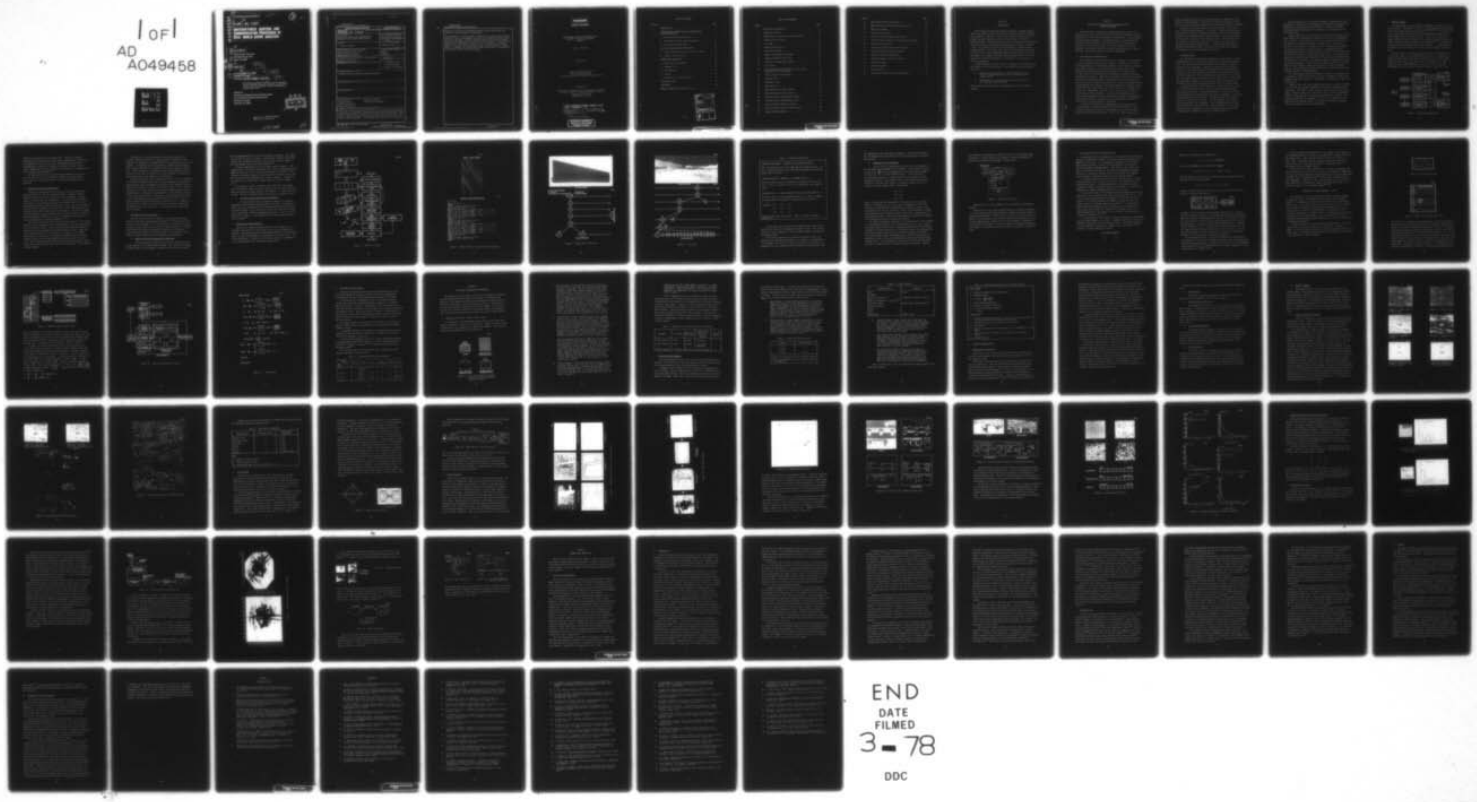
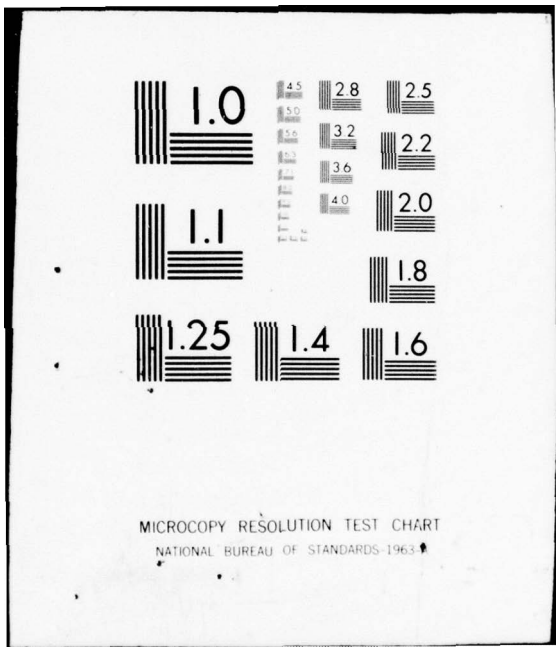
AD-A049 458 HUGHES RESEARCH LABS MALIBU CALIF F/G 9/2
UNSTRUCTURED CONTROL AND COMMUNICATION PROCESSES IN REAL WORLD --ETC(U)
OCT 77 B L BULLOCK F44620-74-C-0054
UNCLASSIFIED HRL-CS-1 AFOSR-TR-77-1303 NL

1 OF 1
AD
A049458



END
DATE
FILMED
3-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(14) HRL Computer Science Report CS-1

(2) 04

(18) AFOSR (19) TR- 77-1303

(6) UNSTRUCTURED CONTROL AND COMMUNICATION PROCESSES IN REAL WORLD SCENE ANALYSIS

(10) Bruce L. Bullock

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, CA 90265

(11) October 1977

(12) 75 p.

(15) Contract F44620-74-C-0054

(16) 2304

(9) Final Scientific Report
For Period 1 April 1974 through 15 August 1977

(17) A2

Prepared For
Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research (AFSC)
Bolling Air Force Base
Washington, DC 20332

Approved for public release;
distribution unlimited.

172 600

1/B

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 77 - 1303	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) UNSTRUCTURED CONTROL AND COMMUNICATION PROCESSES IN REAL WORLD SCENE ANALYSIS		5. TYPE OF REPORT & PERIOD COVERED Final Scientific Report 1 Apr. 1974-15 Aug. 1977
		6. PERFORMING ORG. REPORT NUMBER CS-1
7. AUTHOR(s) Bruce L. Bullock		8. CONTRACT OR GRANT NUMBER(s) F44620-74-C-0054
9. PERFORMING ORGANIZATION NAME AND ADDRESS Hughes Research Laboratories 3011 Malibu Canyon Road Malibu, California 90265		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling Air Force Base Washington, DC 20332		12. REPORT DATE October 1977 13. NUMBER OF PAGES 81
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS (of this report) UNCLASSIFIED 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited .

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Scene Analysis
Computer Vision
Pattern Recognition
Feature Extraction
Production Systems
Artificial Intelligence

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes research results from a program directed toward developing scene-analysis technology to deal with the problems of scene-analysis organization, control, and low-level feature extraction. The work on organization and control has concentrated on the use of a production system framework. To understand the limitations of production

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

systems in building scene-analysis systems, both a high-level and low-level vision system were implemented.

The performance of these systems is described in terms of ease of modification, ease of programming, and ability to embed appropriate domain-dependent knowledge. Several low-level feature-extraction techniques have also been developed concurrently on this program. These include an evaluation of edge operators, use of texture statistics, a glancing operator based on texture properties, and a representation for range motion. A summary is included that places these results in perspective with current work in scene analysis.

HUGHES

HUGHES AIRCRAFT COMPANY
RESEARCH LABORATORIES

UNSTRUCTURED CONTROL AND COMMUNICATION PROCESSES IN REAL WORLD SCENE ANALYSIS

Bruce L. Bullock

October 1977

Contract F44620-74-C-0054
Final Scientific Report
For Period 1 April 1974 through 15 August 1977

Prepared for

Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research (AFSC)
Bolling Air Force Base
Washington, DC 20332

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

INTRODUCTION

Since 1970, the Exploratory Studies Department of Hughes Research Laboratories (HRL) has been conducting an extensive research program in scene analysis. Since 1973, much of the theoretical portions of this program have been supported by the Air Force Office of Scientific Research (AFOSR). The long-term goal of this program has been to develop technology that can derive useful information from complex real-world scenes. The emphasis has been on the development of complete scene-analysis systems. Previously, most work in the field had concentrated on artificial or greatly simplified imagery and had usually led to the development of piecemeal algorithms that contributed little to the construction of practical systems and, consequently, to the solution of real-world problems.

The HRL program is unique in that it attempts to deal directly with the problems of real-scene systems. The primary areas of development have been:

- Evaluation and development of system organization and control concepts based on the use of pattern-directed control rules.
- Development of low-level image analysis operators for use in outdoor scene analysis.

This report reviews the primary accomplishments from this research program.

SECTION 2

SCENE ANALYSIS ORGANIZATION AND CONTROL USING PRODUCTION SYSTEMS

Recent interest in production systems has motivated their use, or potential use, as a system control and organization technique in several applications.[1] This section considers one application: the construction of scene-analysis programs. The general issues concerning production systems and scene analysis will be discussed first to describe the suitability of production systems as a control framework for scene analysis. The specific details of several implementations will then be described with conclusions drawn from their performance.

A. SCENE ANALYSIS CHARACTERISTICS

Scene analysis may be loosely defined as a process for interpreting a scene to produce a description or decision. Programs used for this have invariably used a three-stage paradigm: (1) the image is segmented into subsets relevant to the problem, (2) the subsets have labels assigned to them that symbolically approximate their meaning, and (3) the labels (or scene model) are interpreted to produce the desired description or decision. Applying this paradigm in practice has involved splitting the labelling process into several steps; this has been necessary to provide interpretation flexibility for arbitrary shapes, sizes, viewing angles, and contexts. The simplest example is the blocks world linear hierarchy, which progresses from "lowest level" to "highest level" as follows: edge-points, lines and curves, intersections, surfaces, objects, and scene descriptions. A particularly important aspect of scene analysis programs, and one that directly affects the applicability of production systems, is differences in the processing necessary at these levels (or, more generally, intervals).

In addition to splitting the segmentation and labelling process into intervals, it has become common for the actual topology of the conceptual intervals to have no correspondence to the flow of control between their associated processes. Even in the simple blocks world

linear modelling hierarchy, it has become common for components to have arbitrary interconnection.[2] The ability of production systems to implement or enhance the desired structure and interconnection will be discussed.

Historically, there have been two distinct phases in the development of scene-analysis programs. In the first, concern was with blocks world scenes in which the lighting is uniform, surfaces are nontextured, and objects are rectangular parallelapiped shapes. In the second (and current) phase, outdoor or other complex scenes are dealt with in which the lighting is nonuniform, surfaces are textured, and the objects usually have much more complex shapes. A primary difference between programs constructed for these two phases is the amount and complexity of knowledge that must be embedded in the system.

1. The Blocks World

The knowledge in blocks world programs was derived from a linearly embedded model that was reflected topologically in the system organization. Edge detection is almost always the first (and most primitive) operation on raw image data. Intuitively, edge detection can then be viewed as a "low-level" operation, with higher levels corresponding to the distance one progresses from processing raw image data and toward symbolic information. For the blocks world domain, the levels consist of: edge points, lines, vertices, surfaces, and objects, in that order. This embedding of models is necessary to provide the interpretation flexibility for scenes of arbitrary shapes, sizes, and viewing angle.

Although the blocks world programs all maintain this linear ordering of model levels, the flow of control in such programs has had a great deal more variety. The system organization of the first blocks world programs had a structure that was directly isomorphic to the linear modeling hierarchy just described. Information flowed in a strictly vertical, or bottom-up, direction. Not surprisingly, this primitive control organization was inadequate. The unavoidable noise, texture, and shadows at the lowest level were easily confused for "real" edge points that were propagated to the top causing failure or incorrect interpretation. Examples of this are described in Ref. 3.

The next generation of blocks world programs, beginning with Falk,[4] attempted to correct this error propagation by using varying degrees of model-driven verification in which the flow of control is top-down. Although definite improvements were possible, the performance was far from being robust.

Another variation on blocks world program control was heterarchy.[2,5] These systems were inherently top down but did not have a preprogrammed flow of control. Procedures at all levels are only invoked when they are needed to accomplish something at a higher level. There is no executive control process. Instead, control is distributed throughout the system such that the procedures can act as independent modules monitoring the addition of new information, instead of waiting until the entire scene is passed up through the various levels. In heterarchical programs, for the first time the flow of control was much different from the topology of the models. This greatly augmented organization allowed more noise tolerance on the part of Shari's system and a great deal more generality for Freuder's system than possible in previous attempts. There is a great similarity between heterarchical systems and production systems that will be discussed later.

For completeness, Kiuper's[6] and Waltz's[7] blocks world programs will be mentioned. Kuiper's work was an attempt to implement a blocks world program using frame concepts.[8] Unfortunately, its simplicity limits the demonstration.

Waltz, on the other hand, demonstrated how local syntactic information could be used to great advantage in efficiently achieving global consistency. The effect of this work has crossed over into the second phase of outdoor scene work in the form of work on similar "relaxation" methods.[9] These methods are valuable and perhaps even required in the complex systems that are emerging. But they do not constitute a new approach way to computer vision. In relation to production systems, they can be viewed as rule selection methods.

2. Outdoors Scenes

The second phase of scene analysis development is devoted to outdoor scene analysis. This phase is now in its early stages and there are thus only a few systems available to talk about.[10,11,12,13,14] Already, however, several significant differences are beginning to emerge. First, the knowledge base is much more complex. Instead of the relatively simple linear ordering of features, there is much more emphasis on horizontal variety, or multiple sources of information.[11,13] An example of a simple horizontal organization is shown in Figure 1. These sources of information can include image data from several wavelengths and range data.

Second, there is a dramatic shift in the view of segmentation (one of the three stages of the paradigm mentioned earlier). Segmentation had previously required that labels completely cover the input image space, and usually only one type of label was assigned to an edge. The demand for whole image segmentation has been one of the principal stumbling blocks in every vision system, because in practice it can

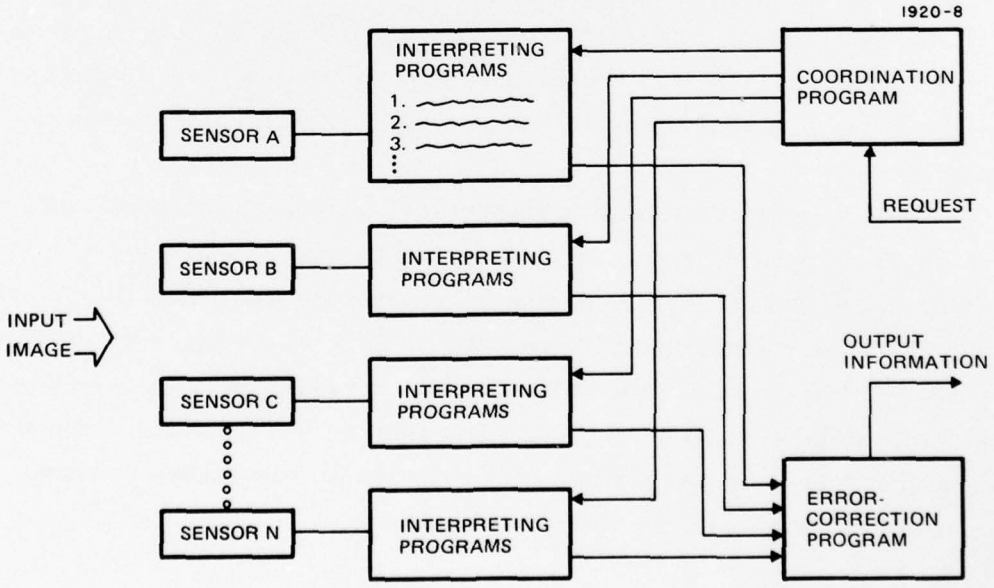


Figure 1. Horizontal organization.

seldom be achieved even in the blocks world. Instead of complete segmentation, an alternative is point feature segmentation. Point feature segmentation can be defined as a nonhomogeneous placement of a nonhomogeneous collection of features to represent a scene. Point feature segmentation has been justified on the grounds of redundancy and used in some outdoor systems.[11,15]

The applicability of the production system framework to the construction and control of scene analysis systems is discussed below. First, however, a few characteristics of production systems will be mentioned.

3. Production System Characterization

The general characteristics of production systems have been summarized by Davis.[16] From his characterization, it appears that two elements are most important in relation to scene analysis: "limited channel of interaction" and "modularity." The limited channel implies a restriction on the interaction between rules because there is no communication other than through the data base. Thus, there is only indirect interaction when subsequent rules must "read" traces left behind in the data base rather than calling other rules directly. Attempts to "kludge" calling mechanisms by sending private tags through the public channel are usually considered contrary to the spirit of the production system concept, although there are notable exceptions.[17,18]

This limited interaction has several important effects.[16] Production systems focus on variations within a domain rather than the common threads that link different facts. Thus, unlike procedural systems, production systems are ideal for domains that characteristically have a large number of distinct states that are difficult to organize. The limited interaction also facilitates a mechanism for global control since any production can fire at any time depending on the contents of the data base. Thus, production systems have a "large scope of attention," which allows them to handle great detail while still being able to react quickly to small changes.

Modularity is the second property of production systems that strongly influences their use in scene analysis. Modularity is the property of a program to be changed without affecting other parts of the program. In production systems, modularity is pushed near its limit, with a single statement line (condition-action pair) being the modular unit. Each statement is an independent chunk of knowledge that has no control over the flow of control to the next statement. The control is determined solely by the contents of the data base.

Modularity provides several important benefits. First, in appropriate problem environments where there are many independent subproblems, high modularity makes programming easy because each statement captures a single action based on a particular data base context.[16] The concept of modularity is of course familiar from software engineering as a means of allowing better construction and maintenance of large software systems. Second, modularity provides a consistent, unified structure since there is only one statement type, the pattern-action rule.[16,19] This uniformity simplifies system modification, interaction with a common rule interpreter to all parts of the system, and, potentially, examination and modification of the system's rule data base since they are easily machine readable.

4. Suitability for Scene Analysis

Thus far there are very few examples of scene-analysis systems actually constructed using production systems.[20] Two systems were constructed on this program. One of these deals with higher level vision, and one with the construction of low-level analysis operators. Several conclusions from the two scene-analysis examples and related non-scene production systems are discussed below. This discussion is preceeded by a brief discussion of the implementation experience.

a. High-Level System Implementation Experience

The two systems we completed were built to explore very different aspects of the scene-analysis problem. The first was an attempt to embed higher level knowledge in production rules; the second dealt

with the implementation of "low level" primitive operators. The higher level system was also an attempt at constructing a system to deal with outdoor scene problems rather than block scenes. For that reason, it was bootstrapped from two existing systems.[21,22]

The basic organization of this system is shown in Figure 2. The capabilities of this system were quite crude. The scene-analysis portion segmented the scene into a tree structure similar to Krakauer's[23] that preserved the spatial relation, size, and area of portions of the scene with uniform texture homogeneity,[21] as shown in Figure 3. Examples of the analysis for a simple and a complex scene are shown in Figures 4 and 5.

The production rules in the high level system were simple graph rules that derived simple relation information from the tree structure model in response to simple queries. For example: (2 large long objects in SKY). The only responses possible were (yes at locations) or (no).

b. Low-Level System Implementation Experience

The second system was an attempt to determine if low-level operations could be written using production systems. We attempted to replace the scene analyzer portion of theprevious system with a production based analyzer. The rules in this system were limited to strings rather than graphs. Without explaining the details, a set of rules are shown below in Table 1, for an operator that locates smooth objects in outdoor scenes.[12]

5. Discussion of Characteristics

Based on this system-building experience, there appears to be a dividing line between the construction of low-level and higher-level programs. Although both involve embedding knowledge into the production rules, the type of knowledge is very different at the two levels. The crucial difference is linked to an observation about the decomposition of a knowledge domain into independent subproblems.[16]

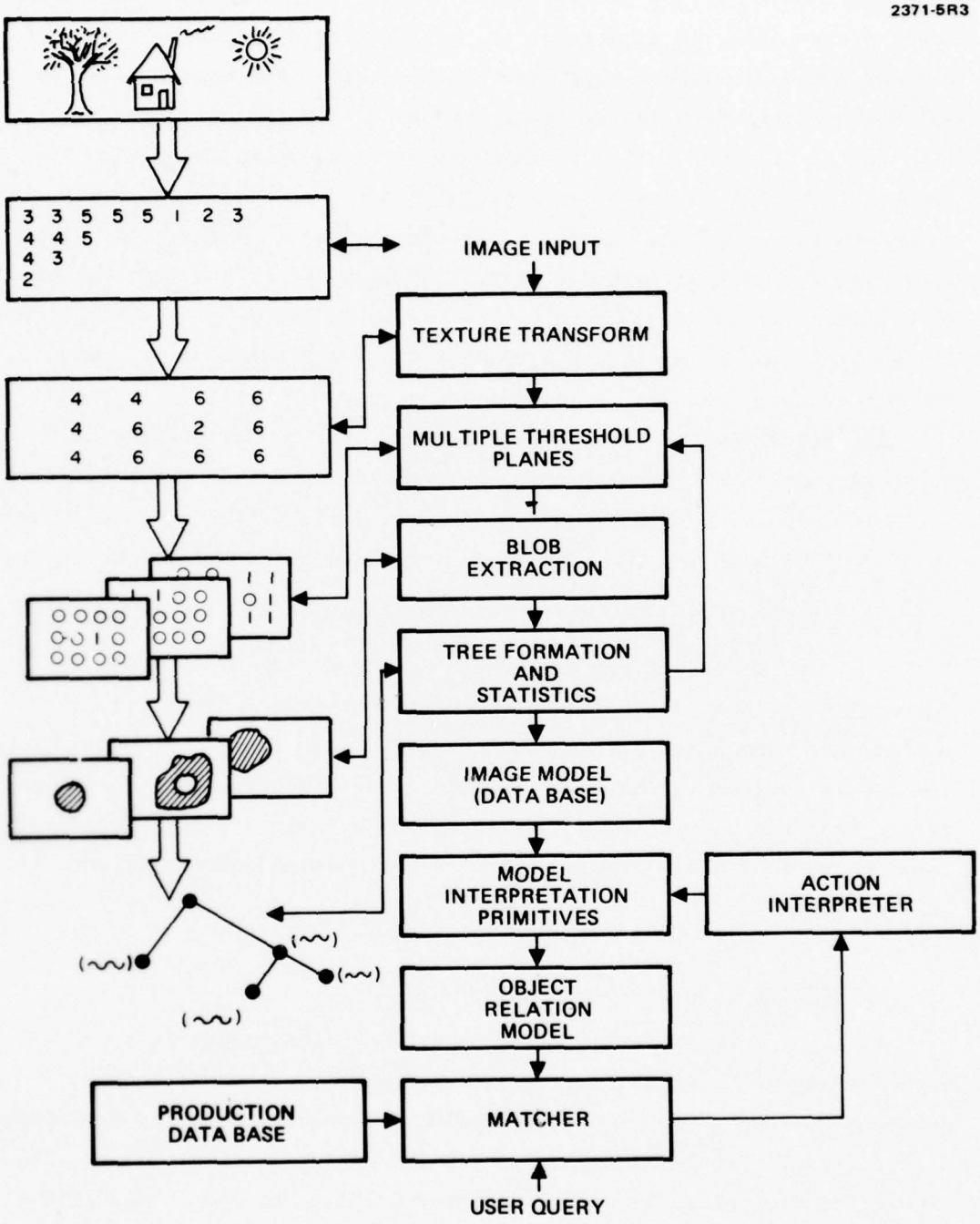


Figure 2. High-level system.

2371-8

SMALL INPUT IMAGE

2371-9

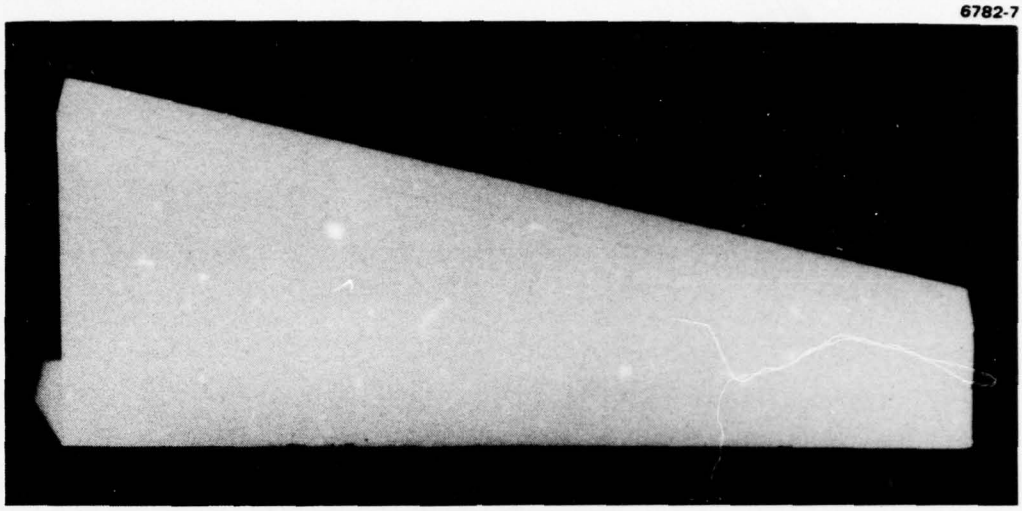
DERIVED TREE DESCRIPTION

```
TYPE SD1.DAT

(MERGED BLOB AT THR 9.)
(BLOB 5.)
(XCOORD 0. YCOORD 2. AREA 2557. AT THR 9.)
(MIN 118. MAX 216. AVGRAY 181.76574)
(MERGING BLOBS AT THR 4.)
(BLOB (0. 2. 1241.) AT THR 4. COMPARES TO BLOB 1. (0. 2. 1241.))
(MIN 169. MAX 216. AVGRAY 196.71555)
(BLOB (31. 2. 13.) AT THR 4. COMPARES TO BLOB 3. (31. 2. 13.))
(MIN 173. MAX 181. AVGRAY 176.69230)
(BLOB (38. 1. 304.) AT THR 4. COMPARES TO BLOB 4. (38. 1. 304.))
(MIN 118. MAX 143. AVGRAY 130.42105)
(MERGED BLOB AT THR 14.)
(BLOB 8.)
(XCOORD 0. YCOORD 2. AREA 3858. AT THR 14.)
(MIN 59. MAX 216. AVGRAY 157.58709)
(MERGING BLOBS AT THR 9.)
(BLOB (0. 2. 2557.) AT THR 9. COMPARES TO BLOB 5. (0. 2. 2557.))
(MIN 118. MAX 216. AVGRAY 181.76574)
(BLOB (0. 18. 730.) AT THR 9. COMPARES TO BLOB 2. (0. 18. 730.))
(MIN 59. MAX 136. AVGRAY 79.903219)
(BLOB (22. 34. 41.) AT THR 9. COMPARES TO BLOB 6. (22. 34. 41.))
(MIN 125. MAX 163. AVGRAY 143.34146)
(BLOB (30. 38. 14.) AT THR 9. COMPARES TO BLOB 7. (30. 38. 14.))
(MIN 152. MAX 180. AVGRAY 163.21428)
(MERGED BLOB AT THR 29.)
(BLOB 8.)
(XCOORD 0. YCOORD 2. AREA 4092. AT THR 29.)
(MIN 59. MAX 216. AVGRAY 156.22849)
ENDOFILE
```

Figure 3. Sample texture region tree data structures.

6782-7



ORIGINAL IMAGE

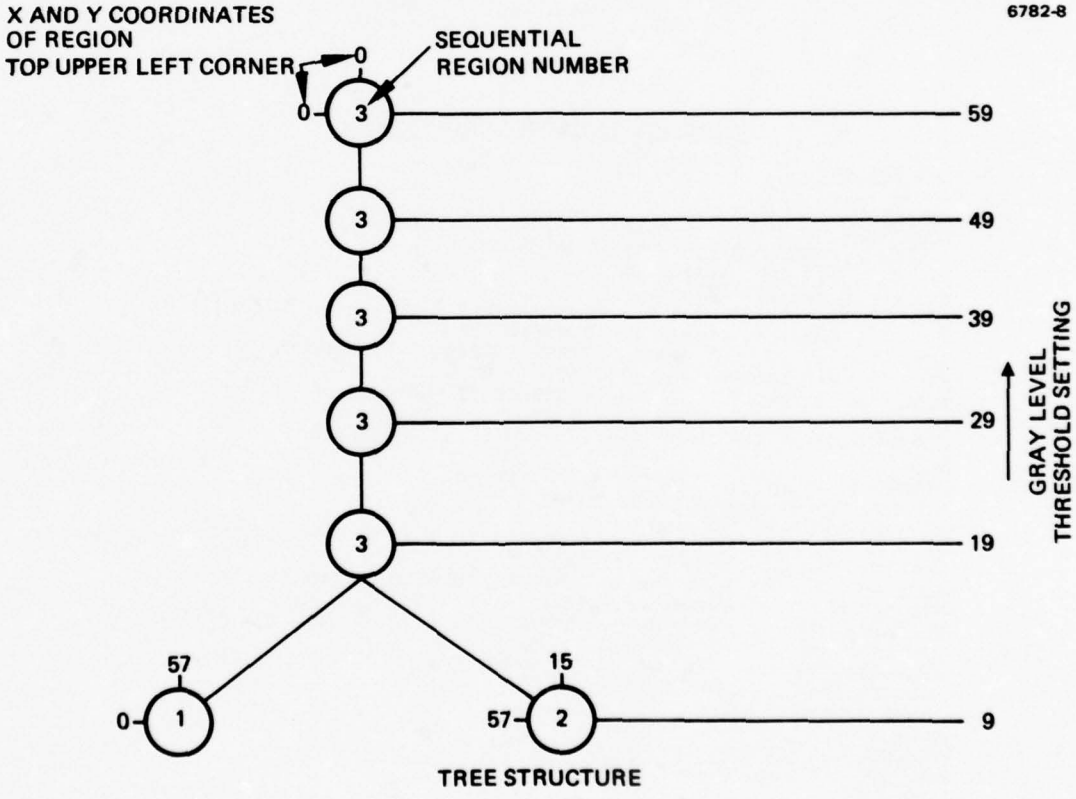
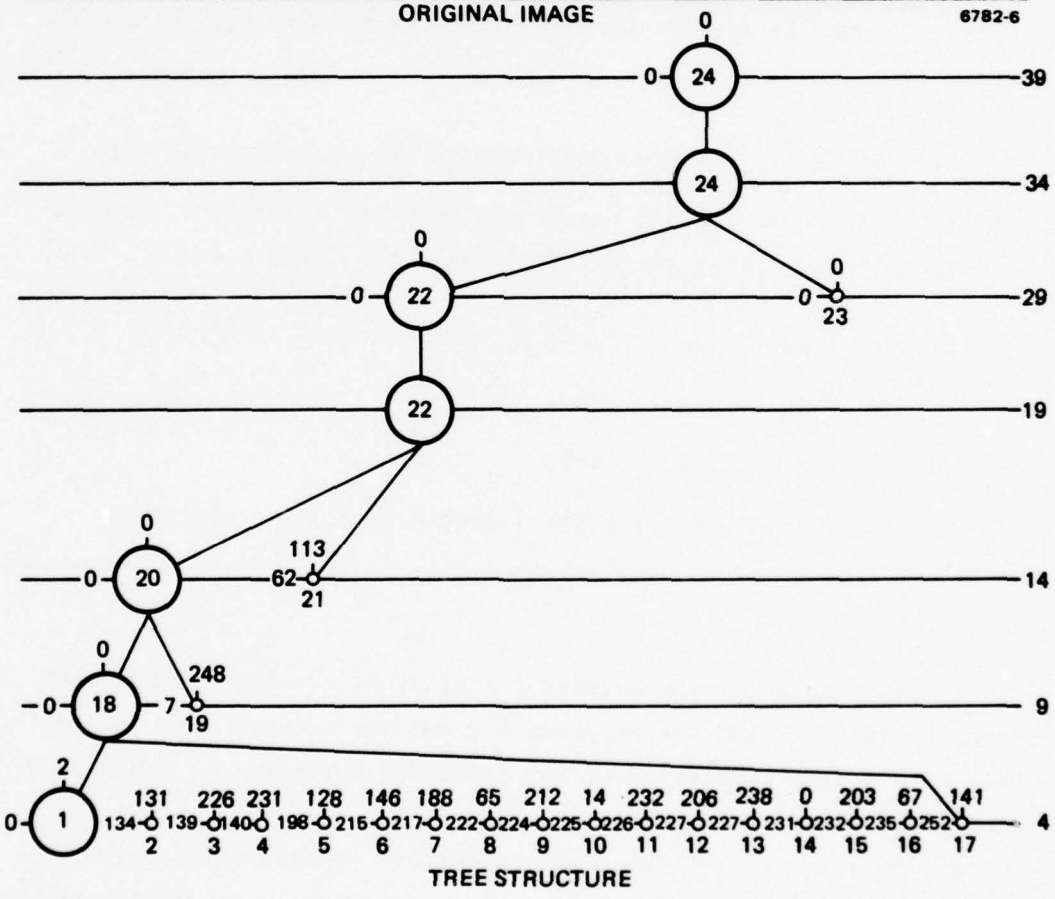


Figure 4. Simple tree block scene.

6782-5

ORIGINAL IMAGE

6782-6



TREE STRUCTURE

Figure 5. Car scene.

Table 1. Low-Level Productions

((IMAGE NOT WINDOWED) — (WINDOW @ 10% AND MARK STATE Ø))

This rule checks to see if the image has been windowed before. If not, it marks the entire scene with window boundaries at intervals spaced 10% of the size.

((WINDOW STATE Ø) → (WINDOW 3 x 3 AND MARK STATE 1))

This rule looks to see if a window has been processed (state Ø if not). If not, it divides it into 3 x 3 subwindows, each marked in state 1.

((WINDOW STATE 1) → (APPLY MOMENT-OF-INERTIA AND MARK STATE 2))

This rule looks to see any subwindows that are in state 1, applies a texture measure, and denotes its application with state 2.

```
((WINDOWS  (W.A.    W.B.    W.C.
            W.D.    W.X.    W.E.
            W.F.    W.G.    W.H.
AND X MAX SET (A,B,C,D,E,F,G,H)) → MARK W.X. STATE 3 AND DRAW
DISPLAY X))
```

A second split exists between the programs constructed for most blocks world tasks and the programs for outdoor scenes. Here the difference is in the complexity of the knowledge necessary to understand the problem domain.

The relative capabilities of production systems in these two domains can be described by looking at five issues: the complexity of the knowledge, the form of the rules, the form of the data base, the globalness of view, tradeoffs between productions and procedures, and

the complexity of the production rule matcher. The first three issues will be discussed in detail below, while the others will be touched on only briefly.

a. Complexity of the Knowledge

A basic issue in constructing the system is the complexity of the knowledge being embedded into the program. A very simple view will be used here. First, the knowledge used in the blocks world programs is structured into a linearly embedded model. On the other hand, the knowledge necessary for outdoor scenes does not possess the same convenient linearity. The control pattern in the linear blocks world systems has also been quite simple. A schematic example of a set of rules for a simple linear system is shown below:

(A) → (B)

(B) → (C)

(D) → (E)

Even if a heterarchical organization is desired, the rule set remains simple. This means that it is very easy to experiment with the construction of such systems by using production systems to control their interconnection,[3] but also that there is very little advantage to adding the production system interpreter rather than using a conventional procedural specification. Early prejudices against using production systems have probably been based on similar observations.

The tight coupling and linearity in the previous model is not present in the knowledge for outdoor and related complex scenes. This is due to the richer problem domain and the consequent greater variety of problems. Although not written explicitly as a production system, the sophisticated office scene system constructed at the Stanford Research Institute[11] uses many isolated chunks of knowledge which could be easily written as production rules. The system specification of Baird and Kelley[24] shows similar rules. The road detectors constructed by Bajcsy and Tavakoli[26] can also be viewed as rules to construct specific operators. Finally,

the footprint rules proposed by Bullock[15] for use in interpreting range information in outdoor scenes is an example of complex knowledge that nicely fits the rule based paradigm, as shown below:

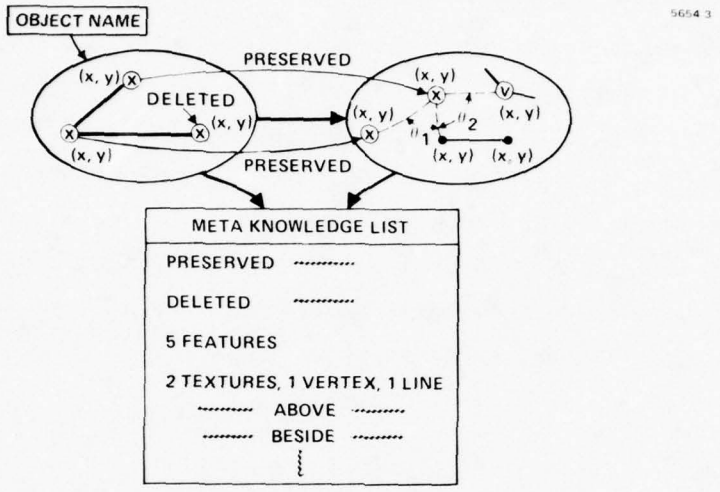


Figure 6. Range footprint rule.

Obviously, all of these examples can be constructed as production systems.

Another view of the potential applicability of production systems is in the transition from model-matching to hypothesis-driven systems.[5,6] Model-matching systems, in which there is usually a very simple class of objects to be understood, can be easily constructed procedurally; but as the knowledge gets more complex and the choices greater, a hypothesis-driven system is necessary. Because the amount of knowledge chunking is much higher in the hypothesis-driven systems, the production systems are ideal.

B. RULE SYNTAX AND DATA BASE ORGANIZATION

An important factor contributing to the successful use of production systems in an application domain is the ease with which knowledge can be mapped into the rules. It is fundamentally important to have a good match between the level of detail in the primitives in the problem domain and program (language) domain. The desire to facilitate such a match has motivated the creation of high-level programming languages. Similarly, in production systems, Davis has noted that the primitive actions should be conceptual primitives in the problem domain.[16]

A secondary factor that directly affects the efficiency of a given production system with a given rule syntax is the organization of the associated data base. The matching process in the production system interpreter becomes increasingly complex and perverse if the data is not organized in a manner topologically similar to the rules.

In most such systems, the information has been represented in rules that were a list structure and the data base has also been a list. This is not particularly surprising since many of the problems have been "verbal." The notable exception is *DENDRAL*, in which the rules use a graph structure to represent molecular structure.[26] A lesser known system used graph production rules to represent the interconnection of input-output (I-O) devices.[22] Although the VIPS system dealt with visual information of type similar to that found in scene analysis, the organization remained a list.[17]

In scene analysis, the concept of spatial relationship is inseparable from the problem domain at all levels. Examples from both low-level and high-level operations will be briefly described. The Roberts gradient, or cross operator, is perhaps the simplest low-level operator. This operator is usually defined as follows:

FOR POINTS ARRANGED

A. .B

C. .D

The point A is defined as an edge point if

$$|A + D| - |B + C| > \text{THRESHOLD} \quad .$$

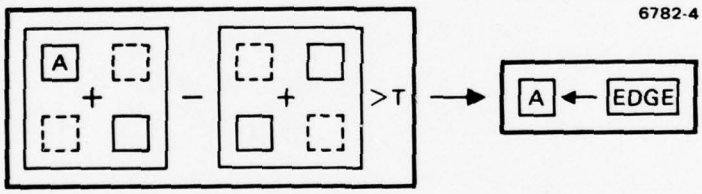
This can be implemented in a procedural language:

$$\text{IF } |A + D| - |B + C| > T \text{ , THEN } A \leftarrow \text{EDGE} \quad .$$

The same operator can be written in production form using a linear list structure in much the same manner:

$$(|A + D| - |B + C| > T) \rightarrow (A \leftarrow \text{EDGE}) \quad .$$

Finally, a graph structure can be used to more closely match the representation of spatial information:



Although it is not obvious that using graph productions for low-level operators simplifies operator construction, good notation would probably make their function and debugging more obvious. There would, of course, be a penalty paid in the form of increased complexity of the associated matching process. The type of graph matching necessary has a computational complexity of $O(n^2)$. Because there are many data points at the raw image data level (a typical image may contain 512 x 512 = $\sim 3 \times 10^5$ primitive matching locations), this type of operation has seldom been attempted for routine use in contemporary processors.

The problems that arise in constructing higher level knowledge are quite similar to the low-level examples just given, with three exceptions: a greatly reduced data base, an increase in the ability to chunk knowledge into single rules, and the possibility that the data can approach a verbal string level at the higher levels.

There is usually a great reduction in the amount of data contained in the representations of a scene at the higher model levels than at the raw picture level. This reduction could be as much as a factor of 100 to 1000. This means that the matching processes that were inefficient at the lower levels may only require a reasonable amount of processing time at the higher levels.

There is also a distinct difference in the types of knowledge that must be encoded at the two levels. As shown in the Robert's example above, the rules are really encoding a tightly coupled procedure that approaches a "kludge" level of implementation.[16] Higher level knowledge is much more independent. Finally, the data base knowledge at the higher levels often approaches an English-text string level that allows rules to be written as lists in the traditional manner[3,11,25]:

((LARGE BLUE) & (ABOVE GROUND)) → (SKY)

The transition to string information implies a separate (perhaps multiple) data base. An analogous situation exists in the HEARSAY system.[27] A disadvantage of this is that the data base is essentially partitioned, as are the rules that can operate at each level. Although this violates the spirit of the production philosophy of giving all rules access to all data in the data base, it corresponds to the partitioning found useful in semantic nets.[20] Our experience has been that this reduces the naturally global "scope of attention"[16] and tends to introduce "private message passing" mechanisms to bridge the gap between the data-base partitions.

A partial solution to this problem has been developed. At the raw image level the data base consists of a pixel array, as shown in Figure 7, that is an exact pictorial (nonsymbolic) representation of the input image. The partitioned data base at an intermediate level is then a list structure as shown in Figure 8.

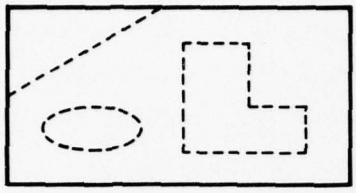


Figure 7. Pictorial data base

6782-3

V	1	VERTEX DESCRIPTOR
V	2	
V	3	
V	4	
	⋮	
R	1	REGION
L	1	LINES
L	2	
	⋮	
L	7	

Figure 8. Higher level symbolic data base.

The solution is to merge these representations, keeping everything in a pictorial format and eliminating the separate symbolic list structure, similar to the recently proposed "symbolic pixel array."[28] In spirit, this pictorial structure should be implemented by actually writing the discovered information into the image (by drawing lines, etc.). It is more practical, however, to use a collection of tags on the pixel array words to denote the data type (intensity, pixel value, edge point, confirmed edge point, vertex, curve, etc.) and then have pointers to the descriptors. An example is shown in Figure 9. Following such a pure pictorial implementation can lead to an interesting implementation in

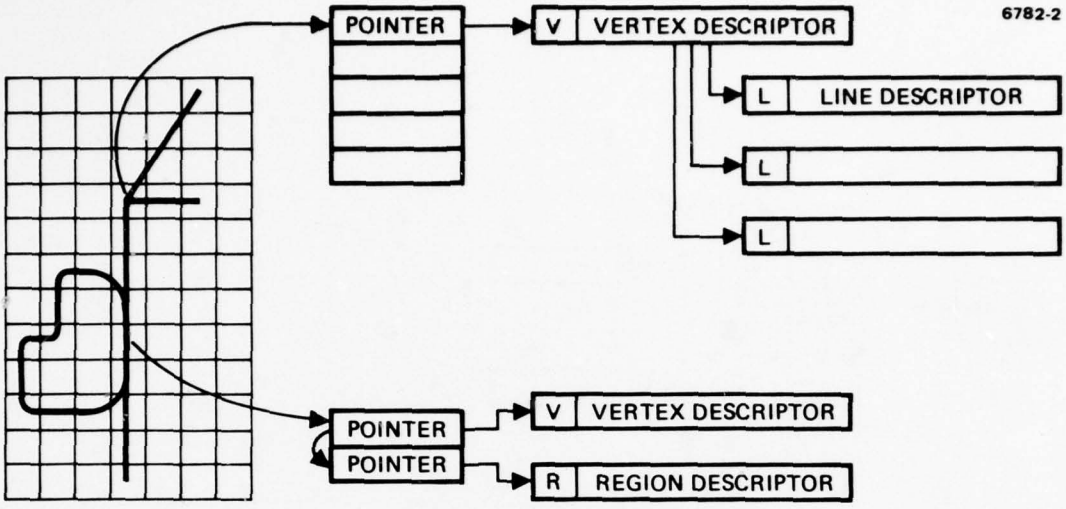


Figure 9. Augmented symbolic pixel array.

which the production rules are also symbolic pixel arrays and the scene analysis capability is recursively used to interpret the input scene data base at many levels rather than at just the original raw intensity level. In such a system, there is a natural pictorial equivalence between program and data. A schematic of such a system is shown in Figure 10.

The effect of this pictorial data base on the rule syntax is to allow single rules to be written pictorially that uniformly access many levels of information all with a uniform topology. An example somewhat in the spirit of Smalltalk[29] is shown in Figure 11. Several symbols need to be defined for use in theprogram. An image subwindow is represented by a square □, and scanning the window is shown by →. A line is shown schematically in a window ⧄. A surface intensity assignment is made Ⓐ. The intensity onto sides of a line is thus Ⓐ / Ⓑ. An angle assignment is shown /α. Simple predicates on the symbols can also be specified.

If Ⓐ = Ⓑ = Ⓒ → LINE •——•

If Ⓐ = Ⓑ →EDGE <<<<<

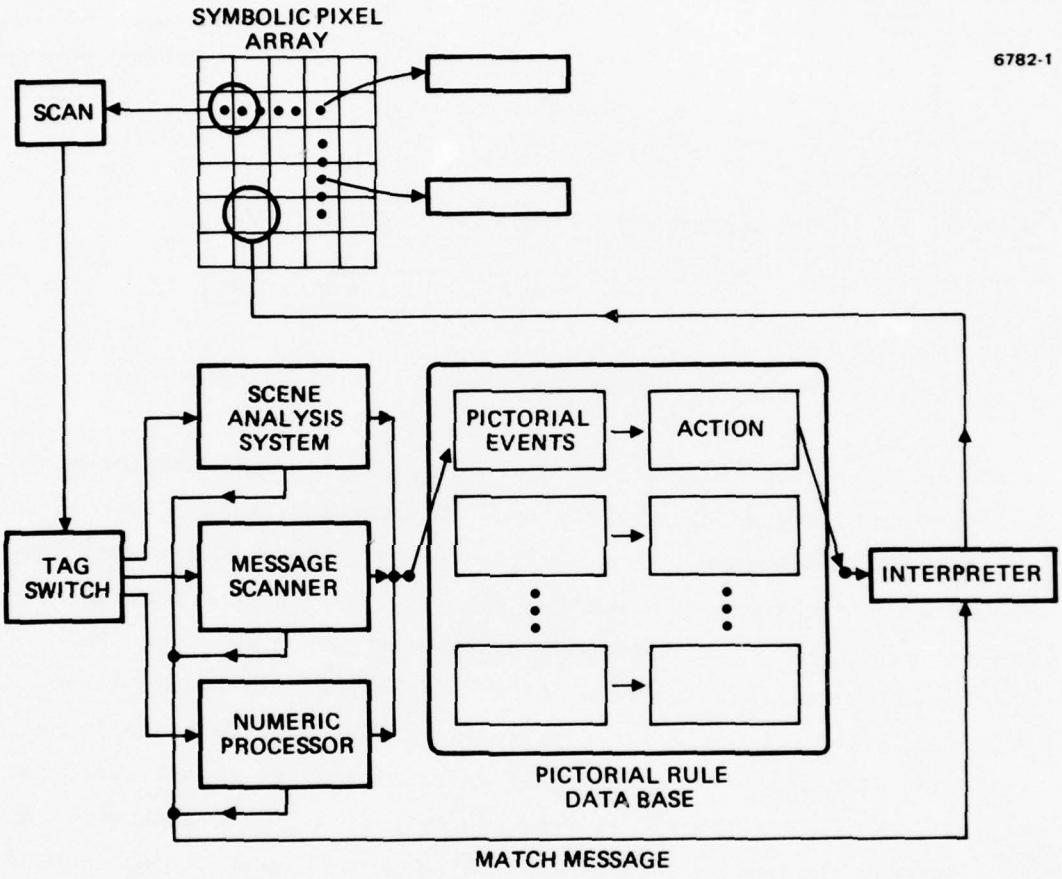


Figure 10. Pictorial interpretation system.

5568-1

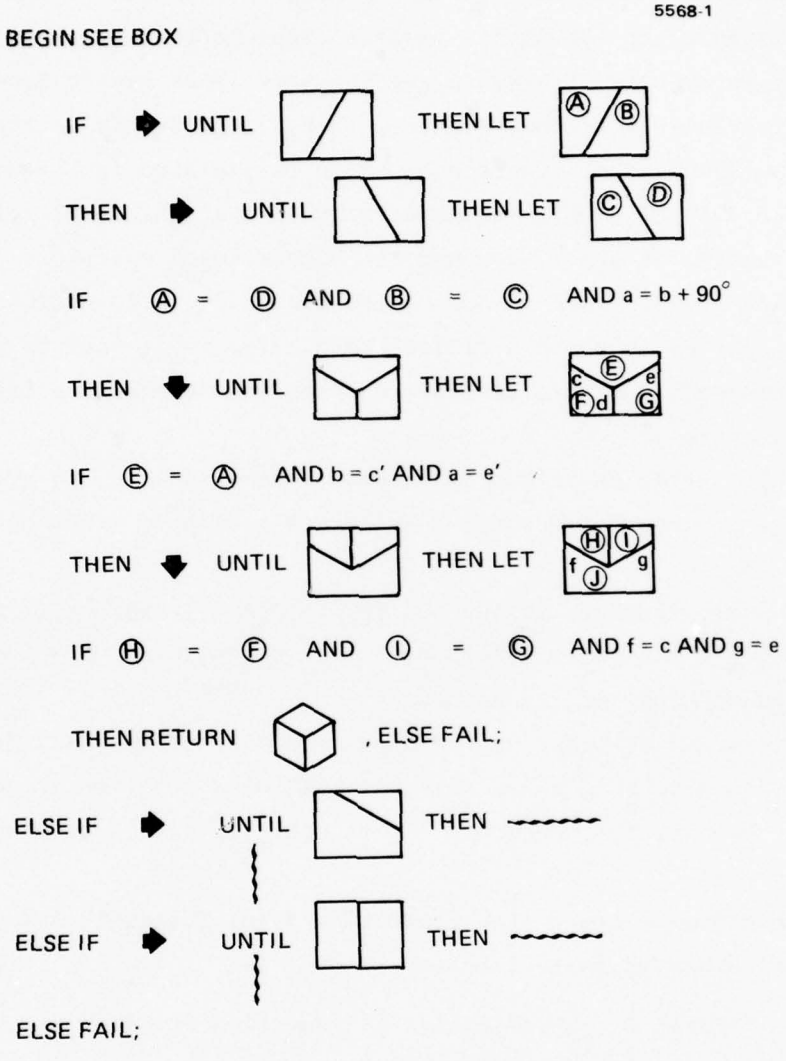


Figure 11. Box program.

C. CONCLUSION AND EXTENSIONS

The general conclusion from the material presented here is that production systems can provide a powerful vehicle for implementing scene-analysis systems if several guiding principles are followed.

If efficiency is a consideration, then, in presently available systems, the lower level operations should be impleted in hardware. There should also be a split in uniformity so that the low-level operators are written as procedures and the higher level operators are written as production rules. This is not unlike the split in traditional complier construction in which a finite-state automata is used to parse the lexical items, while a more general context-free acceptor is used to parse the syntax.

One major strength of the production system idea is its ability to provide a global control mechanism while still keeping track of large amounts of detail.

A secondary strength is the ability to form (graph) rules that can uniformly access the pictorial information in a manner that directly reflects the topology of the scene.

A serious disadvantage is the lack of a clear organizing mechanism to group production rule units together, as found in alternatives such as frames,[8] beings,[30] or actors.[31] This could be partly overcome by using meta-rules.[32]

The relative merits of the production scene framework for scene analysis are shown in Table 2 below.

Table 2. Scene Analysis Production Systems

Production System Characteristics / Scene Analysis Domain	Need for More Structuring	Model Characteristics	Rule Independence	Need for Direct Rule Intercommunication	Current Efficiency	Rule Ordering
Simple low level	No	Spatial relations Scanning Repetitive	Low	High	Low	Grouped
Simple high level	Yes	Relations Structure	High	Modest	Low	Yes
Blocks world	Yes	Relations Structural	High High	Low Low	Low Low	Impossible Impossible
Outdoor scenes	Yes	Numerical				

SECTION 3

LOW-LEVEL SCENE-ANALYSIS OPERATIONS

The objective of this contract has been to investigate the problem of system organization and control for realistic, real-world scene analysis. The task of extracting and analyzing useful image features is the vital first stage of every scene-analysis system. Unfortunately, it is also very difficult, especially for outdoor imagery. This section describes the results obtained on this program towards the implementation of useful feature extraction operators for outdoor image analysis.

A. GENERAL FEATURE TYPES — POINT, LOCAL, GLOBAL

An examination of image features shows that they fall into three general categories: point, local, and global. Figure 12 shows a representation for each category for a simple scene example. The trade-offs between these categories are discussed below.

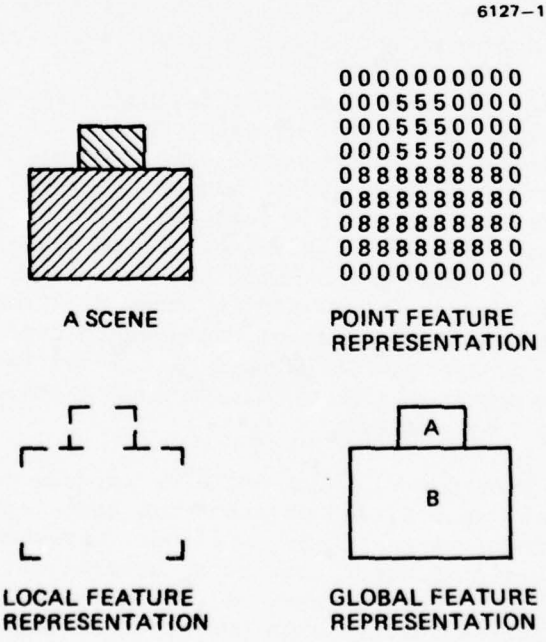


Figure 12. Scene representation through the point, local, and global types of features.

- Point Features — Point features are used to represent a scene as a matrix of values for every resolution element or pixel in the image. The point value for each pixel represents either the intensity magnitude (a function of the reflectivity or emissivity) or the range from the sensor to the point. Figure 12 shows a matrix of values that represent the pixel intensities. Point measures have the advantage that they are usually available directly from the image sensors with little additional processing required for their extraction. Their major disadvantage is that they have poor invariance characteristics. For example, they can vary widely with small changes in illumination and contrast levels. It is sometimes possible to perform transformations on the point feature data to overcome the lack of invariance but, as a rule, these transformations are computationally very complex.

- Local Features — Local-feature measures include average intensity over an area, locally connected line segments and curves, and line and curve intersections. Features based on local measures have greatly improved invariance characteristics in comparison with point measures. These invariance characteristics arise from local averaging and the use relative measures, as in the detection of edges. The presence of a line segment, for example, will not change for a wide range of illumination and viewing angle changes, even though the absolute values of the point features producing the gradient may shift dramatically.

  The point features represent an image exactly, although with little invariance or data compression efficiency. Local features, on the other hand, represent an image in an abbreviated or abstract manner. The relative positions and orientations of line segments and line intersections, for example, may be sufficient to specify an object's shape. An image model using local features has the advantage of greater invariance to image differences and a smaller memory requirement compared to that for point feature matching. An example of a local feature model is also shown in Figure 12. In this example, the local features are corners.

- Global Features — Global features include regions, entire surfaces, shapes, and objects that have been segmented or extracted from an image. A global representation or model for a building might consist of several rectangles connected in a specific way. A trivial global representation of a block structure with two separate regions A and B is shown in Figure 12.

Global features have a high degree of invariance to image differences. Unfortunately, global features are the most difficult to extract successfully. This is because they depend on the segmentation of complete regions or surfaces from the scene.

Table 3 summarizes the above discussion on image features. This table shows that the point features suffer from poor invariance characteristics and, therefore, are inappropriate as a primary component in outdoor scene models. Further, it shows that global features are in general more difficult to extract, but can provide better invariance when available. Feature extraction methods that successfully identify both local and global features have been developed. Based on this qualitative comparison of feature characteristics, the feature categories are given an approximate utility ranking that can be used in a control utility function.

Table 3. Comparison of Feature Types for Scene Models

Category	Invariance	Extraction Difficulty	Transform to Correct for Invariance Errors	Relative Utility
Point features	Poor	Trivial	Difficult	0
Local features	Good	Moderate	Not always necessary	1
Global features	Excellent	Difficult	Not necessary	2

1. Generic Feature Examples

As briefly mentioned above, there are many local and global scene features. This section will discuss a large collection of features that have a high potential for use in modeling outdoor scenes.

Although the point features are, by themselves, inappropriate for use as features, they do supply the basic data for the identification of local and global features. Most local features are based on the use of edges in the image. These can be derived by detecting discontinuities

in the point feature data. In a dual sense, many global features are derived from uniform regions in the scene found by propagating the similarity of some property within a region rather than the difference across a boundary. Because these two feature types are fundamental, edges and regions are the basis for most useful scene features.

- Edge Features — The discovery and analysis of edge point data leads naturally to the development of line and curve segments and vertices at the intersections of line segments. Measurements can then be performed to produce "derived features" in the form of relative lengths, angles, number of lines meeting at a vertex, vertex locations, and endpoint locations. The primary local and derived features are listed in Table 4. The utility values are based on their associated degrees of freedom.

- Region Features — Global regions are apparent in a scene as areas of uniform point feature values (patch of uniform reflectance, texture, or color). Because a region has a boundary it also forms edge points that can be analyzed as curves or piecewise line segments. From the region's boundary points and interior area points, many derived measures can be formed to characterize the region. Several of these are listed in Table 5.

Table 4. Locally Derivable Features

Feature	Utility Value
Line[a]	2
Curve[b]	3
Vertex	2n + 2

[a]Line is not considered to have definite length

[b]Curve approximated by short, straight segments for simplicity.

Table 5. Global Features

Global Feature	Comments
Area	∿Size
Perimeter	
Area/4π* (perimeter)^2	Closeness of region to circle
Radius of gyration $\lambda = (\mu_{20}+\mu_{02})^{1/2}$	
Invariant moments	Unique shape characterization
Centroid position	
Length	
Width	
Length/width	Aspect ratio

- Texture Features — It has already been shown that the unreliable point measures can be used to determine edges and regions. In addition, the region surfaces frequently have texture properties that can be measured to derive feature information. Statistical texture features have received the most attention. They are derived from the surface gray-level histogram.

  First-order (mean, variance, skew, kurtosis) and second-order measures (energy, entropy, correlation, moment of inertia) statistical means can be derived from this histogram. Although first-order statistics can be used for relative measurements (such as uniformity), they cannot be used for texture classification because of their sensitivity to scene contrast.

  The second-order statistical measures are based on information about pairs of image points represented in a gray level dependency matrix. These statistics have been shown by Haralick to be invariant to scene contrast if a histogram equalization is performed on the gray-level statistics that describe texture characteristics such as complexity, coarseness, and homogeneity.

Table 6 lists several first- and second-order image statistics that have been evaluated.

Table 6. Statistical Measures Used in Texture Analysis

FIRST ORDER
1. Minimum, maximum, and mean gray values
2. Histogram peaks
3. Contrast: $\frac{MAX - MIN}{MAX + MIN}$
4. Skew (histogram symmetry)
5. Kurtosis (histogram flatness)
6. Variance (histogram dispersion)
7. Entropy
SECOND ORDER
8. Angular second moment (amount of edge, related to the energy in the image waveform or the average uncertainty.)
9. Entropy (related to the complexity of the scene)
10. Correlation
11. Angular second moment inverse (related to the homogeneity of the image)
12. Moment of inertia (related to the coarseness of the image texture.)
13. Kikuchi entropy

B. OPERATOR CHARACTERISTICS

1. Edge Operator Evaluation

Quite early in the investigation the performance of several edge operators was evaluated. The results, which were presented in Ref. 33, are summarized below.

Two types of edge detection must be considered for adequate real-world scene analysis. The first was defined as macro-edge detection, which involves major surfaces. The second was microstructure edge detection, which involves boundaries of surface texture elements. Six edge operators were then evaluated to determine their performance at both macro and microstructure analysis. These were thresholding, two types of filters, and the Sobel,[34] Kirsch,[21] and the Hueckel Operator.[35]

The evaluation was made on one traditional blocks world scene and six difficult real-world scenes with texture from several contexts. The performance of each operator was very consistent from scene to scene but, as expected, varied greatly between the operators. The Kirsch and Jueckel operators show the most promise for future use. The Kirsch operator can be viewed as an excellent "conservative" edge operator for real-world scenes. It is very successful at finding the predominant edges in difficult images. The Hueckel operator can be viewed as a "thorough" edge detector. It finds all of the predominant edges, as well as most of the very low contrast edges. Unfortunately, the Hueckel operator is also computationally more expensive.

A useful strategy was suggested to improve the efficiency of real world edge detection. First, the conservative Kirsch operator is applied to find all of the predominant edges. These candidate edges are then a first interpretation of the scene's edge structure. The information can then be interpreted in terms of a model and the user's goal to form a plan for further analysis. The analysis is then carried out by selectively applying the thorough Hueckel operator on the basis of the analysis plan to find more information where needed. This balanced strategy is more efficient than running the Hueckel operator exhaustively and extracting too much detail to efficiently process on a first pass.

Sparse and dense textures are defined in this report on the basis of the available edge resolution. Sparse textures usually have high microstructure edge contrast, while dense textures have less apparent contrast. Edge detectors with low edge contrast sensitivity can usually extract sparse texture microstructure edges, just as they do high contrast surface boundaries. Edge detectors with good sensitivity to low contrast edges are naturally better at extracting dense texture microstructure edges. This was confirmed in the experimental results. Further, the Hueckel operator was shown to be the most sensitive to low contrast, dense texture microstructures.

Specific conclusions about each of the six operators are summarized below.

- Thresholding

Gray-level thresholding produces the poor results expected because of the slow gradients in the original images.

- Preprocessing by Filtering

The examples show that very little is gained by preprocessing natural scenes. If prefiltering is done, the problems of noise and possible information loss should be carefully considered. Most of the high-frequency information enhanced by the Laplacian can also be extracted by either the Kirsch or Hueckel operators alone. Although not attempted here, local high-pass filtering to enhance microstructure edges may prove useful.

- Sobel Gradient Operator

This simple gradient operator is shown to be very insensitive to low contrast gradients and gradients that have textual unhomogeneity. This results in very poor capability to extract dense texture microstructure and internal surface edges over which there is little contrast. However, high contrast surface boundaries and sparse texture microstructure can be reliably extracted.

- Kirsch Operator

The Kirsch operator is slightly more complex than the Sobel operator but produces much better results for natural images. The definition of the operator makes it sensitive to texture gradients and to simple uniform brightness gradients. This feature preserves continuity of the detected texture microstructure better than the Hueckel operator and makes relatively good performance possible even for dense texture microstructure.

- Hueckel Operator

The Hueckel operator is by far the most sensitive to low contrast edges. This increased sensitivity is at the expense of speed and computational complexity, however. With a high "DIFF" setting it does an excellent job of extracting major surface boundaries and sparse texture microstructure. The slightly poorer performance but increased speed of the Kirsch operator makes it a better choice for this task, however. The Hueckel operator is most useful when selectively applied, with a low "DIFF" setting, to extract low contrast dense microstructure edges.

- Sample Results (Tank Image)

This is an example of a scene with both a difficult object and background terrain. It has been digitized to the same standards as the two previous aircraft images. This image is more difficult than the other aircraft images because of the complex background features.

The original image is shown in Figure 13. As would be predicted, the low-pass filter (Figure 14) succeeded in covering up some of the dense background microstructure. Also, the high-pass filter result shows greatly enhanced microstructure (Figure 15). Thresholding (Figure 16) produced the expected complex, difficult to interpret result. Because the edges on the tank are fairly distinct, the Sobel operator was reasonably successful (modulo the resolution of the original) at object segmentation (Figure 17). It has, however, mixed all but the most distinct (sparse) background microstructure. The Kirsch operator was more successful at extracting the microstructure (Figure 18). The results of a high-pass followed by a low-pass filter, and vice-versa, are shown in Figures 19 and 20. Neither case particularly helps the situation. The results of the Hueckel operator with a high DIFF setting (Figure 21) are only slightly better than the brightest edges in the Kirsch operator result. The Hueckel operator result with a low DIFF setting (Figure 22) is actually more difficult to interpret than the microstructure in the Kirsch result due to the loss of edge continuity. The Hueckel result, as suggested earlier, may contain finer detail that can be interpreted under the guidance of the Kirsch result.

3688-17

Figure 13. Tank in desert original image.

3688-18

Figure 14. Low-pass filter.

3688-19

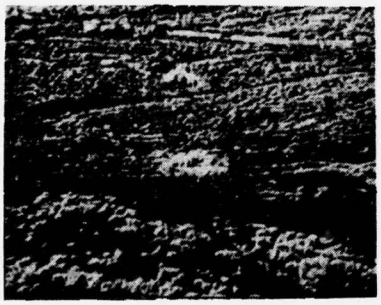


Figure 15. High-pass filter.

3688-20

Figure 16. Threshold at 55.

3688-12

Figure 17. Edge detection by Sobel operator.

3688-13

Figure 18. Edge detection by Kirsch operator.

3688-14

Figure 19. High-pass filter then low-pass filter.

3688-15

Figure 20. Low-pass filter then high-pass filter.

3688-16

Figure 21. Edge detection by Hueckel operator.

3688-11

Figure 22. Hueckel operator with low DIFF setting.

A summary of the relative performance and computational characteristics for the edge operators is shown in Table 7.

Table 7. Edge Detector Comparison

Edge Operator	Performance Rank	Computational Complexity
Roberts cross	4	N (3a)
High-pass filter	4	N (9a)
Laplacian	4	N (9a)
Sobel	3	N (14a)
Kirsch	2	N (72a)
Hueckel	1	54 (a+m) = 270a

Key

N = Number of image elements

a = Machine add cycle time

m = Machine multiply cycle time (assume m $\sim$ 4a)

2. Line Finding

Lines can be an important local shape characteristic of edges associated with objects (especially man-made) and context detail. Unfortunately, the basic edge operators described in the edge detection section do not determine if there is any structure in the collection of edge points they detect. Therefore, a different method must be used to associate structure on a collection of detected edge points.

There are many approaches to the structure-finding problem, including line and curve fitting, dynamic programming, heuristic search, and the transform techniques.[34] When the goal is to mechanize complete global segmentation in terms of connected boundaries, then the fitting and searching techniques produce good results. Unfortunately, the complexity and noise in outdoor scenes usually makes it impractical to apply global segmentation. Also, these operations are computationally very complex, usually requiring at least m^2 operations, where m is the number of

detected edge points. An attractive alternative is to use transform techniques. Although it might be possible to use the Fourier transform for such a purpose, a much more effective transform is the Hough transform.[36] The Hough transform has been used successfully to find isolated line and curve segments[36,37] and has a complexity of order m.

The basic notion of the Hough transform is to map edge points in the image space into curves in the transform space on the basis of the normal parameterization of the line (curve). A simple example is shown in Figure 23. Concurrent edge points generate curves in the transform space that intersect at a common point corresponding to the slope and y-intercept of the line. The transform space information is deposited as a two-dimensional accumulator array, then the important slopes and intercepts are found by searching for maxima. Finally, possible line intersections are found from the line segment position data by solving sets of simultaneous equations. Thus, the information provided by this process about edge features in the scene is the position, length, and orientation of isolated line segments. For intersecting lines, it provides the position, number of intersecting lines, and their angles. This information is stored in the model as nodes with the correct image space coordinates and with labels on the nodes as to the details of the features.

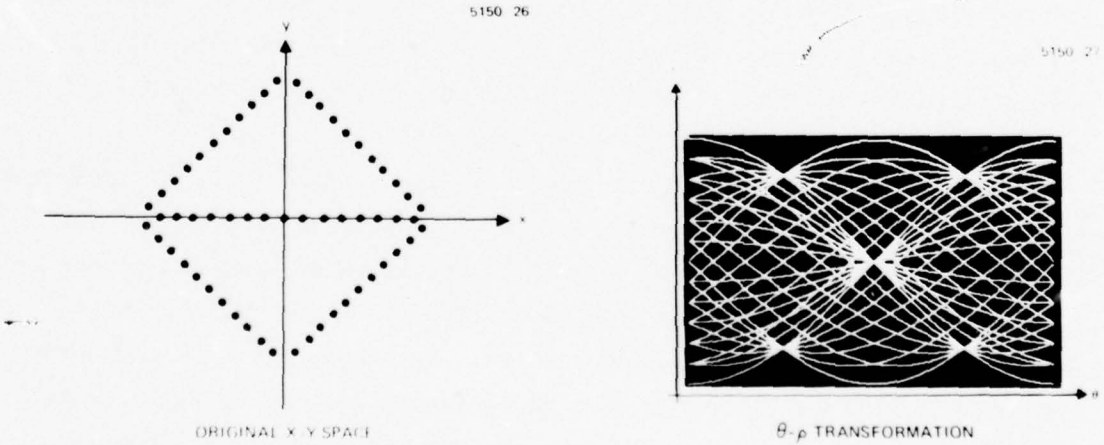


Figure 23. Hough transform example.

Early development of the Hough transform technique for line finding were carried out on this program using the simple scheme shown in Figure 24.

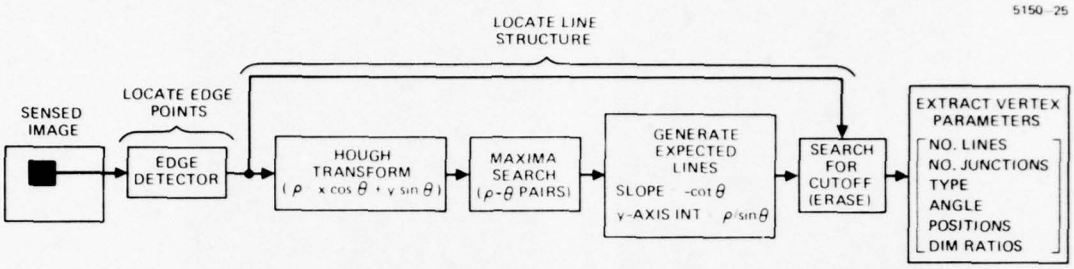


Figure 24. Edge feature extraction process.

The first experimental results are shown below in Figures 25, 26, and 27. These results are described in more detail in Ref. 12.

Because this program emphasized system organization and control, the line-finding process was not developed further. These successful initial results, however, provided the starting point for refinements carried out by A. Luk and S. Dudani on a DARPA contract.[39] Some typical results are shown in Figures 28 and 29.

3. Texture Measures

Texture is an inherent aspect of all real-world visual scenes. The ideal edges and homogeneous surfaces that have been the cornerstone of present vision research exist primarily in images of man made objects. Most real world scenes (outdoor, medical, etc.) present the primitive aspects in the form of textural information, texture edges, gradients, regions, and surfaces. The human ability to deal with this texture information is so well adapted and the processing seems to be done at such a low level that we are seldom aware of the textural characteristics in an image. Unfortunately, for reasons of priorities, lack of understanding, and processing time, texture has been essentially ignored in computer vision. But because many of the important application areas are inherently textural, it is vital that the computer analysis of textures be better understood. The work reported on here is a first attempt

5006-3

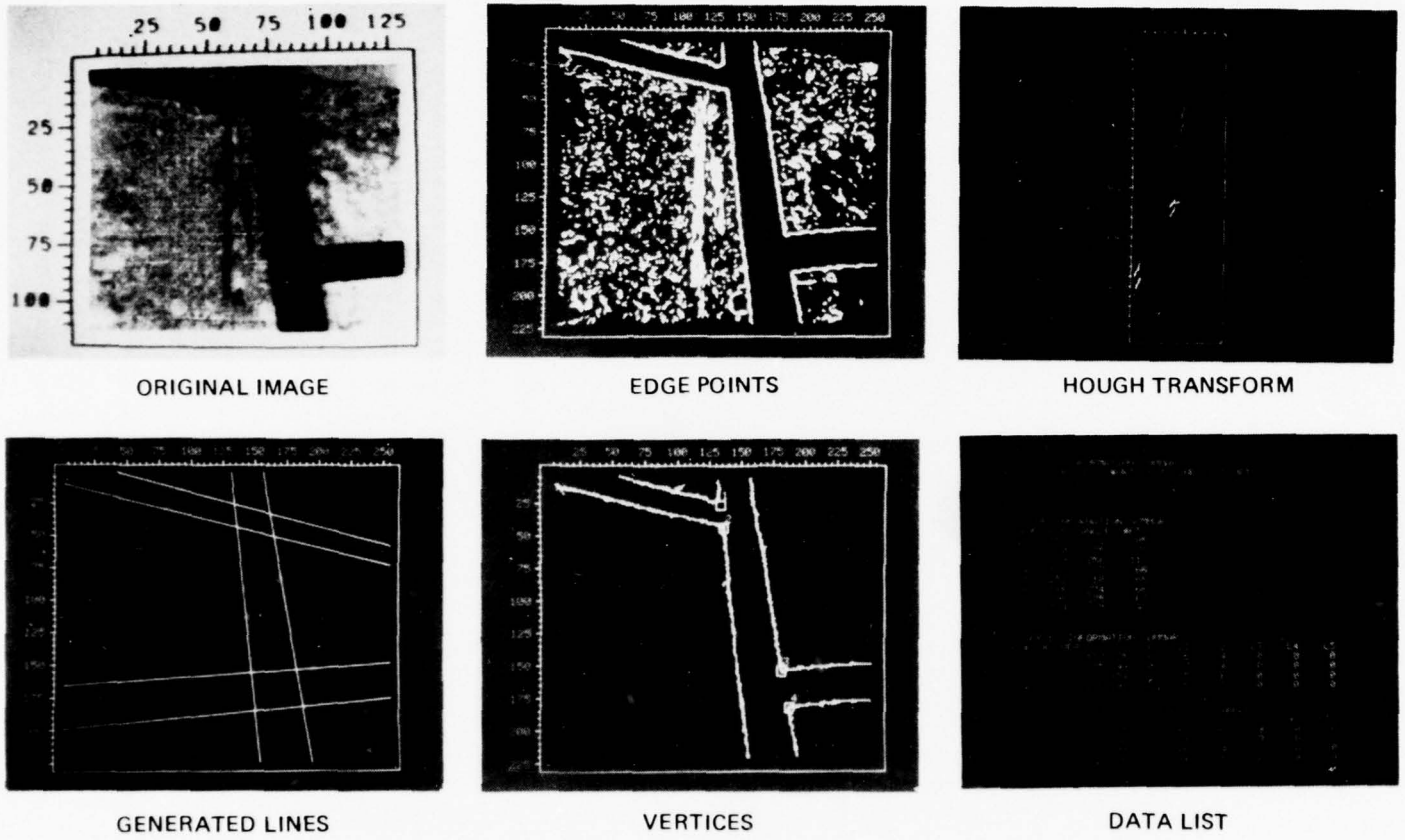


Figure 25. Edge-line-vertex feature extraction process.

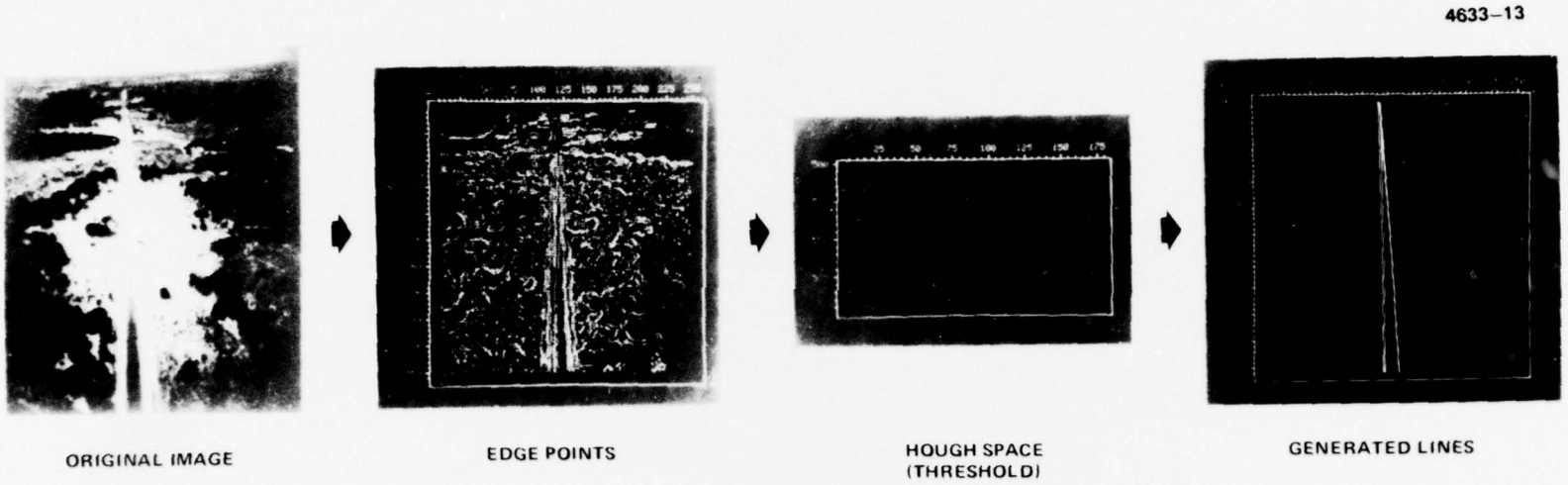


Figure 26. Road line finding.

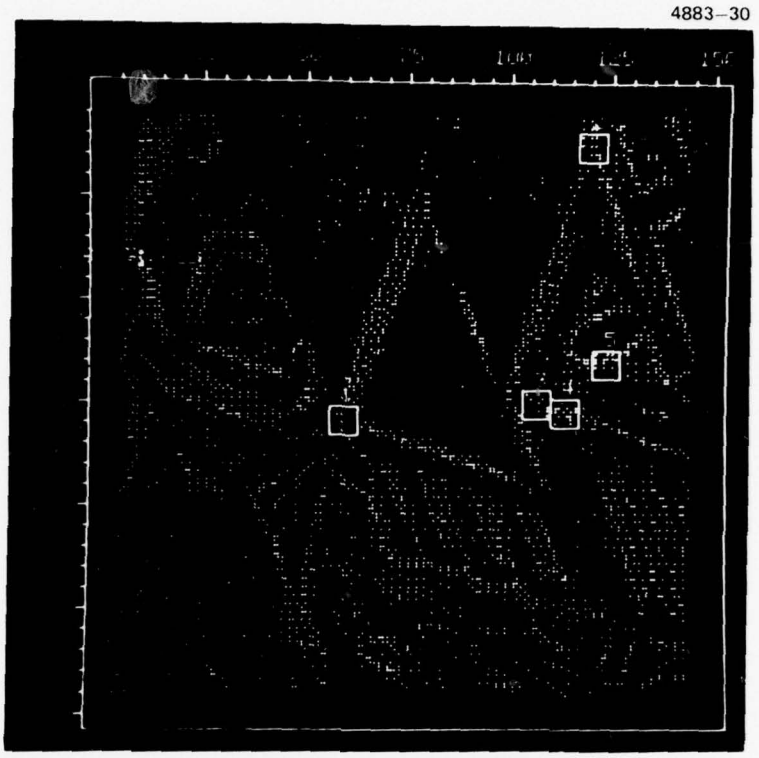


Figure 27. Complex scene vertices.

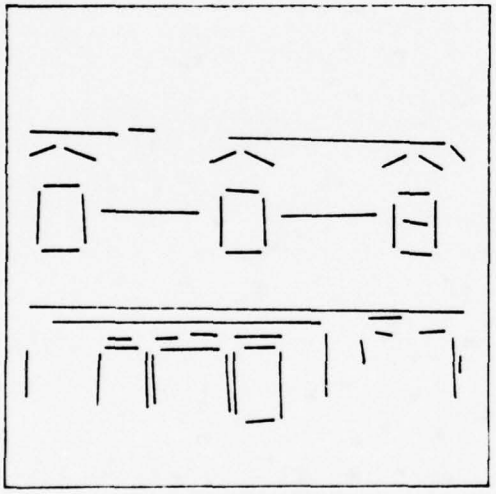
to build a sophisticated texture-analysis system. Previously, there has been some work on low-level statistical analysis of textures, region-growing programs based on brightness and color properties, simple shape analysis, and region growing based on shape regularities. The specific work to be reported on in this report is concerned with the implementation of the low-level statistical aspects of texture analysis and the extension of these techniques to derive low-level directionality and structure information.

The first implementations of the measures shown earlier (in Table 6) used these measures only as a texture transform (transform the input image into a new one where brightness was a function of the texture). Relative comparisons between local areas in an image were then performed to obtain area texture characteristics. Examples of such characteristics (for uniform areas) are shown in Figure 30.

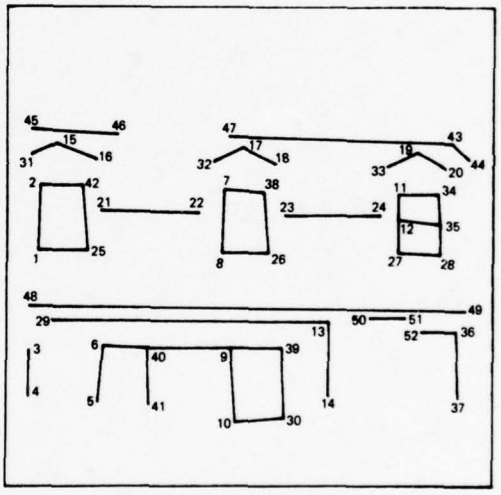
6127–22

SCENE

EDGE ELEMENTS



LINE SEGMENTS



VERTEX MODEL

Figure 28. Line and vertex finding on house scene.

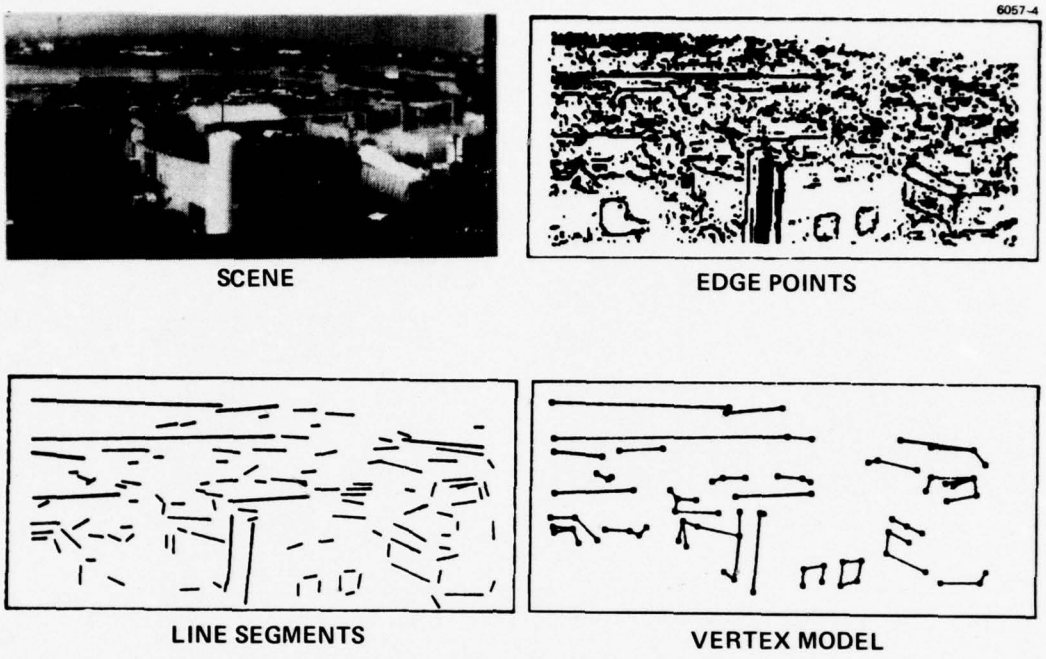
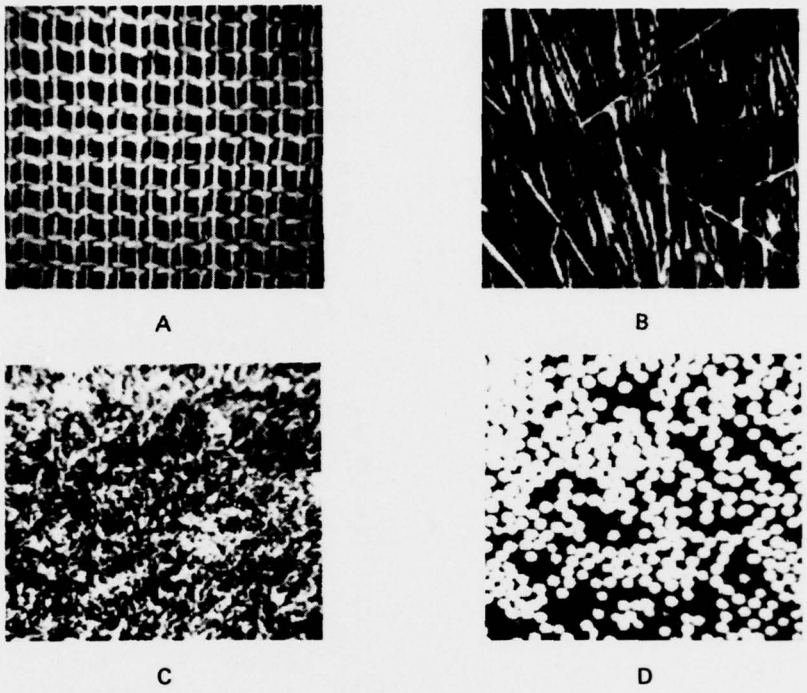


Figure 29. Line and vertex finding on low-resolution building.

These preliminary studies hinted that much more information could be derived from these simple measures. First, by measuring the statistics separately in four directions, information can be obtained on texture directionality. Second, by varying the spacing between pairs of points when the statistics are collected, information can be obtained on the lengths of the texture element.

The results of such a process in one direction are shown in Figure 31. These graphs show the values of the six second-order measures as a function of spacing between the pairs of points. Although the values vary between the measures, the positions of turning points are often the same. The spacings that correspond to the turning points are the dimensions of texture microstructure in the given direction.



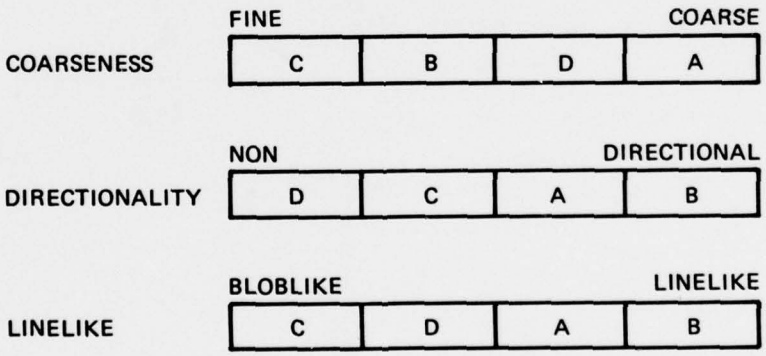


Figure 30. Texture measure results.

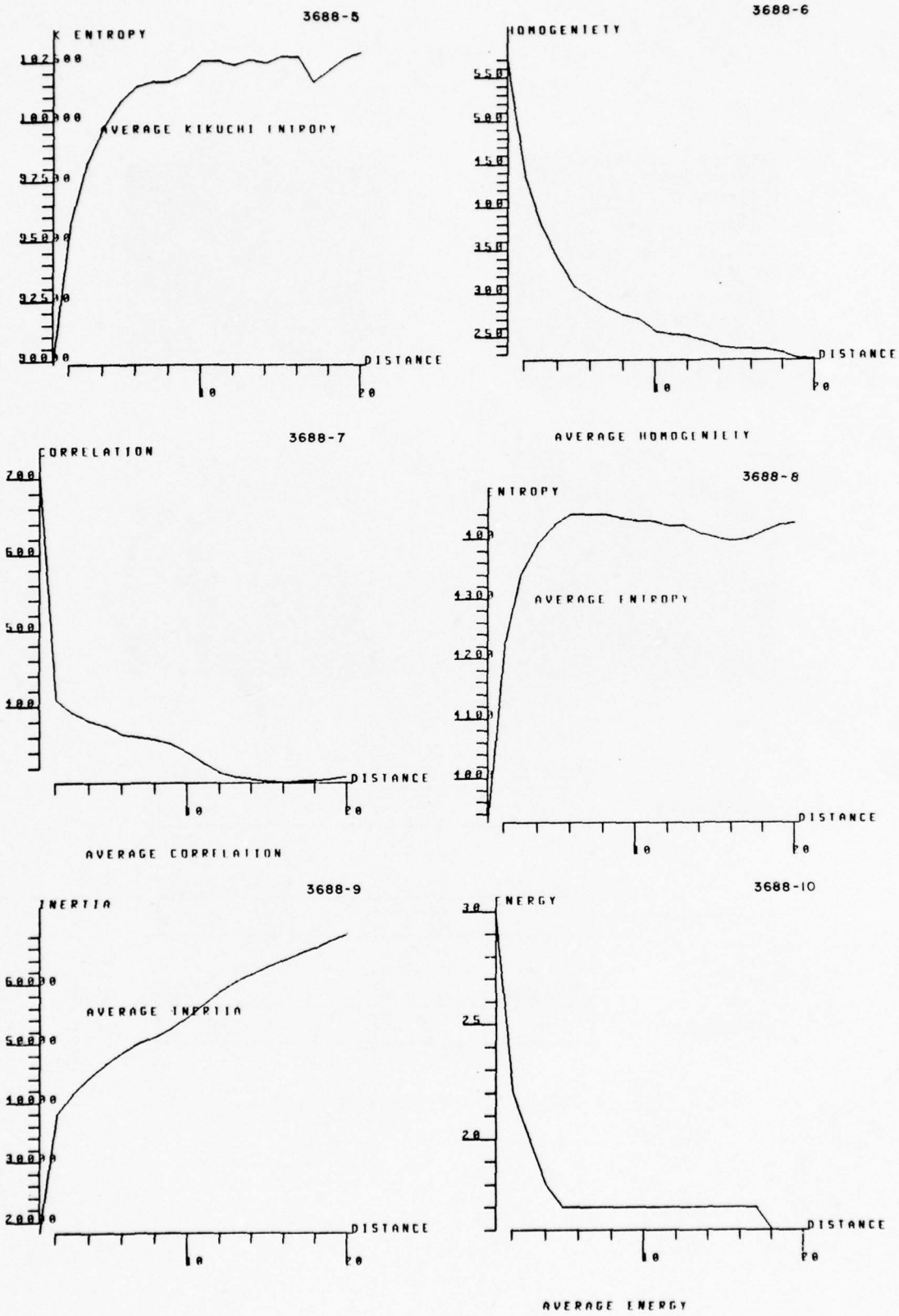


Figure 31. Output data sample from texture analysis.

4. Simplified Texture Directionality Measure

During the edge operator investigation, we realized that the Soble operator could be used to measure edge direction and magnitude. The direction of edges is important information that is usually difficult to extract. For example, the Fourier transform was one of the first texture-analysis techniques because it provided directionality information (at great computational expense, unfortunately). The statistical operations described earlier can be used to derive direction information, but they are also expensive (although much less than the Fourier). Therefore, we decided to investigate the ability of the Sobel operator to determine texture directionality.

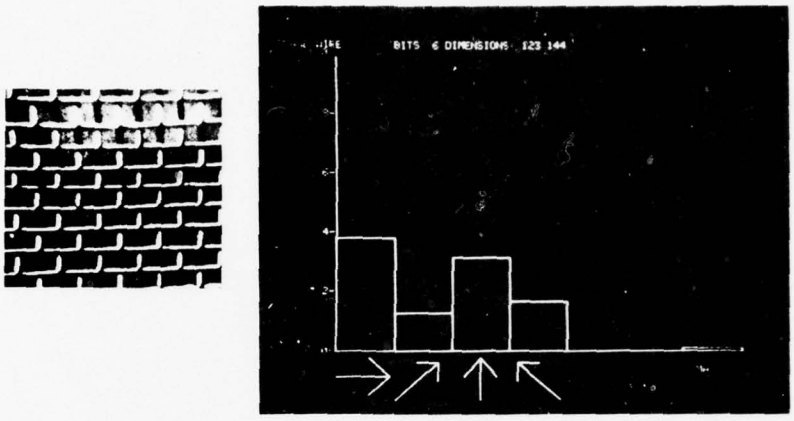
The Sobel operator is described as a 3 x 3 gradient operator that computes a value for each point in the image. The operator about point I with the eight surrounding points labeled

A	B	C
H	I	D
G	F	E

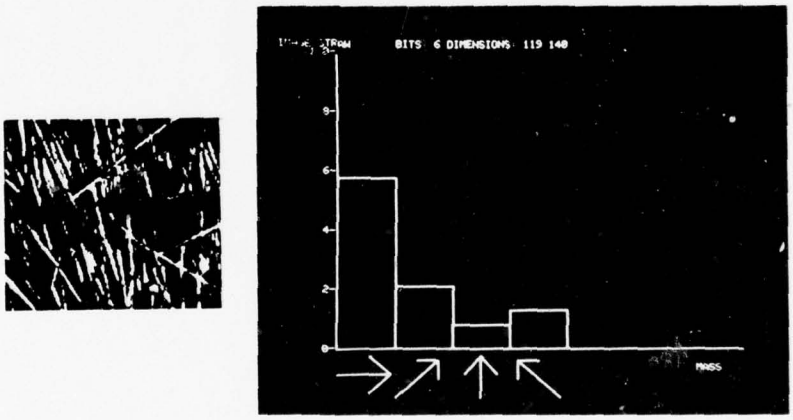
can be described as: $|G| = (X^2 + Y^2)^{1/2}$, where $X = C + E + 2D - G - A - 2H$ and $Y = C + A + 2B - G - E - 2F$. The X and Y components can then be used to determine the edge direction, which in this case was quantized. This method was successful. The results in four quantized directions are shown in the histograms in Figure 32.

5. Low-Resolution Features

A difficult task that often arises in outdoor imagery is the analysis of a distant scene with little resolution. There may be too little detail for the edge, line, and texture features, developed thus far, to be found. Therefore, we attempted to determine features that might be useful in analyzing such imagery.



(a) Angle histogram plot (quantized in four directions) showing horizontal and vertical predominances in "wire" image.



(b) Horizontal tendency in "straw" image (note that arrows show direction of edge gradient)

Figure 32. Angle histogram experimental results.

An early result was an operator called the interest operator.[12,38] This operator evolved from work on texture analysis discussed earlier. The motivation for this operator was to greatly reduce the portion of a large low-resolution scene that would otherwise have to be examined in detail. The computationally simple interest operator can be applied first to find likely areas with some structure, then a more sophisticated operator could be applied for more detailed examination in the small area or to consider the geometric relation of the areas. This strategy keeps the amount of computational effort applied to a scene area proportional to the expectation return from the effort.

Hughes previous work on texture analysis showed that the areas with high structural content will normally have a coarse texture. This suggested that the moment of inertia measured the joint amplitude probability density was appropriate. As it stands, the moment of inertia operator is unable to tell the difference between coarse textures of objects, roads, or terrain. This situation can be improved if it is assumed that there are highly structured areas on man-made target objects and that these areas are spatially unhomogeneous. Roads and ground terrain, on the other hand, are usually spatially homogeneous in at least one direction. Thus, an appropriate interest operator should find regions that are coarse and spatially nonhomogeneous.

Based on the above criteria, the interest operator was implemented as shown in Figure 33. First, the image is partitioned into equal sample windows. A reasonable window size would be one-tenth of a linear dimension of the total image. This assures that the selected windows will be large enough to contain reasonably distinct features for final analysis, but small enough to make the processing time reasonable for the final selection phase.

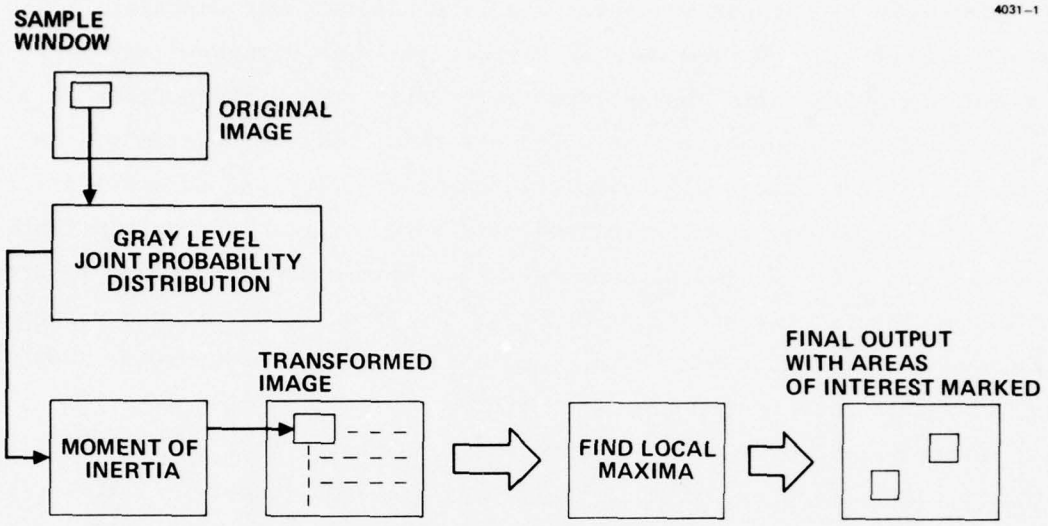


Figure 33. Interest operator mechanization.

Second, the moment of inertia is computed for each sample window. But to get the spatial homogeneity information, the measure is calculated in four directions (horizontal, vertical, and left and right diagonal) at each point. Of the four values thus obtained for each point, only the minimum of the four is retained. This strategy has the property that if the nonhomogeneity is low in any one direction, then the value of the entire measure will be low. Of course, the measure is also low if there is little variation in any of the directions. High values will thus be assigned only to those regions that are highly irregular and likely to be interesting.

Finally, the resulting value for each window is compared with the value of the eight surrounding windows. The window is marked as interesting if and only if its value is greater than the eight surrounding values. This is equivalent to selecting only the local maxima window as interesting. This step enhances the rejection of large uniform areas or surfaces.

An example of its application is shown in Figure 34. This figure also shows that the tagged areas are essentially rotationally invariant.

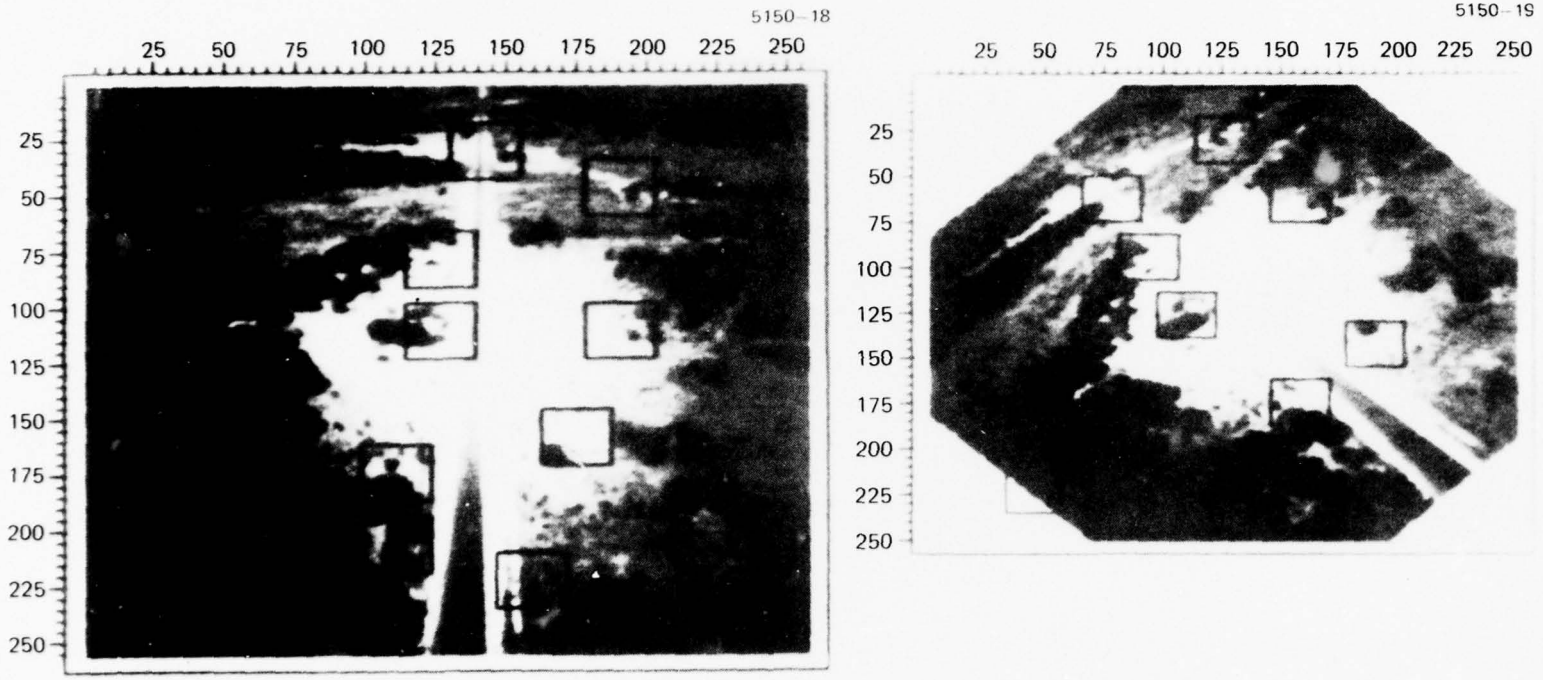


Figure 34. Texture prominence processor results.

A more elaborate low-resolution model, called a "footprint," was later developed using both the texture interest and strong lines (when present). An example of such combined features is shown in Figure 35.

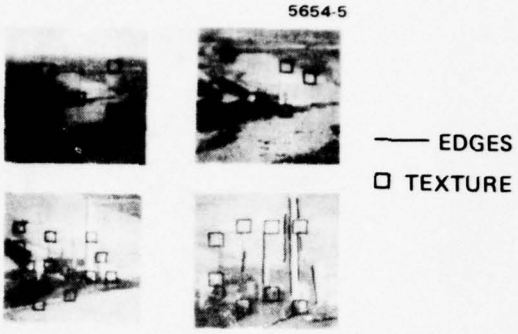


Figure 35. Footprint example.

The four images show the change of these features with respect to distance. A technique for using such distance-related footprints for the analysis of range motion was outlined in Ref. 15. This scheme used a linked model. The link was between a representation at a distant range, and one at a close range, as shown in Figure 36.

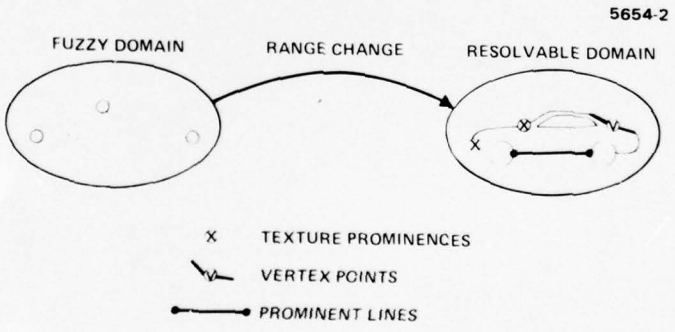


Figure 36. Linked footprints.

This type of linked data structure can be elaborated by procedural attachment to form a frame-like representation that captures meta-knowledge about the expected changes between footprints. An example is shown in Figure 37. The use of such a linked footprint for goal-directed analysis is shown in Figure 38.

5654-3

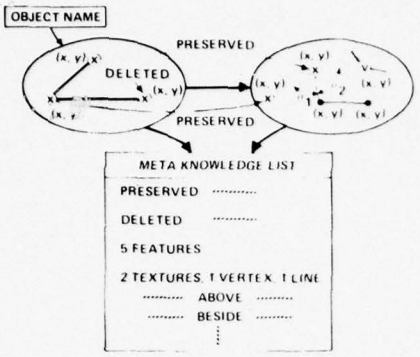


Figure 37. Data structure.

5654-4

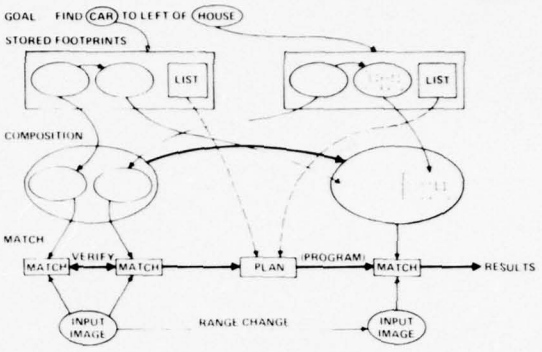


Figure 38. Goal driven analysis and plan composition.

This example shows how a given goal can be used to retrieve the relevant linked footprints, establish a combined low-resolution match and then force a composition of the associated meta-knowledge to form a plan to find the objects at a close range.

SECTION 4

SUMMARY AND PERSPECTIVE

There are three purposes to this summary. First, to briefly review the research progress made on this program. Second, to assess the current problem areas in computer vision as seen from the perspective gained on this program. Third, to suggest areas of particular importance for future research.

A. APPLICATION REQUIREMENTS

Until very recently, scene analysis technology has been limited to simple scenes with a limited variety of simple object shapes, good contrast, and little background or surface texture. Unfortunately, there is a large gap between this capability and the current real-world application requirements in satellite image processing, navigation, industrial automation and inspection, office automation, and medicine.

The requirements for real-world scene analysis usually fall into one of two system categories. The first system, called a "static observer," must deal with complex three-dimensional objects, reflectance variations and shadows, surface and background texture, and partial or missing parts. This type of system has been the primary goal of past scene-analysis research. The second system, called a "dynamic scene narrator," must be able to deal with a wide range of environmental dynamics in addition to possessing all of the same static capabilities. These can include dynamic goal priorities, motion of the viewer or individual parts of the scene, changing resolution, and sometimes unpredictable background context. Although this type of dynamic system is necessary in many applications, it has been given relatively little attention. Many of the aspects mentioned above have been dealt with individually and ad hoc, but no attempt has been made to combine these capabilities into a single system.

With the requirements of the above two categories as a guide, the existing scene-analysis technology can be evaluated for its applicability. The scene-analysis technology will be described in terms of three essential elements: segmentation, representation, and control.

B. SEGMENTATION

Virtually every scene-analysis program has used the segmentation paradigm as the primary tool for establishing order in an image. Segmentation is the process of partitioning an image into subregions that are homogeneous with respect to some feature property. Shape analysis is then usually performed by interpreting the sub-regions and their relationships.

There are at least eight fundamental methodological tools associated with segmentation: contouring, edge detection by template differencing, edge detection by functional approximation, region growing, detection of macro and microstructure using clustered features, guidance from defocused images, guidance from glancing, and utilization of multisensor data. Each of these has independently grown to be quite sophisticated for dealing with different subproblems of segmentation, yet segmentation capability as a whole remains poor for complex outdoor scenes. Significant progress in low-level vision can be made at this point, not by trading off one technique against the other, but by carefully examining how the best independent elements of each can be integrated into a single segmentation process that is better than anything currently available. This has not been achieved in any system, including the existing cluster-based systems, to any degree of generality. Although not an easy problem, it is certainly within the reach of any group willing to carefully implement a system of greater complexity than that any previous segmentation process.

Among the segmentation methods mentioned above, contouring is probably the oldest method. It involves the successive thresholding of the image gray levels into contour planes and then tracking around the resulting area.[40] This technique was abandoned early in block scene analysis because the resulting segmentation had many unwanted fragments due to reflection highlights and surface gradients. Surprisingly, this process has been used to produce segmentations of difficult real-world imagery in two examples.[21,24] This method is not suggested as a replacement for edge or region segmentation, but as a complementary method. It can be used, for example, to find large areas that are interesting on

the basis of a thresholding predicate such as texture or color (the blue sky, for example). A contouring technique was used in the high-level production system example described in Section 3. It successfully demonstrated that such a scheme can be useful in outdoor scenes if correctly applied.

The concentration in early scene-analysis work on the blocks world led to a preoccupation with the second tool, edge points in block scenes. We studied the performance of several edge operators on difficult real-world scenes[33] and found that the Hueckel[35] and Kirsch operators[21] give the best performance on complex scenes. Based on the variation in operator performance, further work on edge operators is important, but it should be done in the specific context of real-world problems. One approach would be to refine the use of existing operators under the control of better interpretation programs.

A common conceptual mistake when people are first introduced to scene-analysis problems is to assume that the segmentation problem has been completed with edge detection. Unfortunately, structure must first be extracted from the noisy edge point locations. Several methods of doing this have been proposed, including curve following and coding,[34,40] curve fitting,[41] and the Hough transform.[34] As with edge detection, many of these solutions have been developed for simple scenes and their behavior on complex imagery is not well known. However, coding techniques are seriously hampered by noise, and the fitting methods are inefficient unless highly goal directed. Moderate success has been achieved with the Hough transform methods on real imagery for finding curves and lines.[37] These techniques have been extended on this program and are described in Section 3.

Unfortunately, the structure-identification methods mentioned above are optimized for regular man-made shapes in which long straight lines and simple curves are predominant. Virtually no work has been done towards developing a general technique for natural structure isolation schemes. For example, it is hard to imagine using any of the existing techniques to find a cloud on the basis of shape.

The fourth tool is a dual process to edge segmentation called region growing.[42] Instead of finding boundaries on the basis of local differences, region growing propagates regions on the basis of a criterion of similarity. Region growing has seen a recent revival of interest after successfully segmenting difficult outdoor scenes.[10] One advantage of region growing over edge detection is that a boundary description is automatically produced, eliminating the need for search or transform techniques. Unfortunately, region growing has its own inefficiencies: the process of merging and splitting regions during propagation is essentially a heuristic search. Claims have been made that region growing is better than edge detection for complex real-world imagery because it can easily use multiparameter information (intensity, color, etc.) and because it is more global and thus less sensitive to noise.[30] In fact, however, edge detection can be done just as easily for multiparameter data, and averaging over windows can be used to attain the same global effect.[43]

Among the tools listed above, glancing and multisensory data are probably the least understood. The idea of glancing at a scene to find interesting areas certainly has merit in outdoor applications. The use of texture and strong edge information for this purpose is described in Section 3. Once such an area is found, it is concentrated on by a structure-analysis process. A variety of such glancing operators, optimized to cue on different aspects, could be used to make a preliminary pass at the scene to determine: the next analysis step, what analysis operators should be applied, and how and where they should be applied.

Connected-object segmentation is the most frequently used approach for organizing scene-analysis systems. There is growing evidence that additional approaches are necessary, particularly in the outdoor scene environment. From an intuitive view, it seems doubtful if the demand for connectedness associated with segmentation is realistic in most outdoor scenes. Further, it seems doubtful that a complete segmentation, or even every edge or even every edge or region, is necessary for understanding outdoor scenes. The idea of using "distinguished"

features[11] from multiple sensors (if possible), is an initial step toward new approaches. These would be an extension from the planning phase to the analysis phase of the glancing, mentioned above. The early stages of an extension using collections of isolated features from multiple sensors, feature locations from glancing "interest" operators, and available "obvious" region and line-intersection data together in one representation is shown in Section 3.

The few exercises so far done with outdoor scenes have shown that color is an extremely powerful cue for segmentation.[10] The importance of color has also been verified in the analysis of LANDSAT photography by the excellent success of simple classification schemes that do not use shape at all.[44] We expect that color will continue to play an important role in real-world scene analysis. There are, however, some unexpected problems. Color in many situations may not mean the usual red, green, and blue bands. Instead, it will probably be more common to have colors widely displaced in the spectrum from ultrasonic, through millimeter waves, and up to ultraviolet. The unsolved problem for such colors is the difference in resolution and the nonregistrability of the imagery.

In addition to color, important cues for three-dimensional segmentation can be obtained from range data. This can be provided by millimeter-wave or laser range finder,[45] by stereo,[46] or by gradients.[47] The significance of range data has been demonstrated by its ability to simplify the segmentation and shape analysis of complex three-dimensional objects.[48] One of its best properties is its invariance to the illumination effects that trouble most edge feature data. The excellent success of these initial experiments shows the importance of further development in these areas.

There has been a growing flurry of work on texture analysis for real-world scenes. There are currently two approaches to texture analysis: statistical and structural. The statistical approach is based on measures such as entropy and the moment of inertia of the gray level co-occurrency matrix (or second-order joint probability distribution).[49] The values derived from these measures can be used

directly for crude classification,[50] as the basis for segmentation,[51] or to derive information about properties such as homogeneity and coarseness,[52] as shown in Section 3. The structural approach is based on the spatial and directional characteristics of the texture. This information can be derived directly[53] from indirect properties of the co-occurrency matrix,[54] the Fourier transform, or simple nontransform histograms, as shown in Section 3.[47,55] Unfortunately, texture has a recursive nature, with the statistical characteristics at the following level. Thus far, no uniform approach has been devised that acknowledges this dual aspect. Segmentation in real-world scenes, the understanding of surface types, and the analysis of background context will be aided with further progress in texture analysis and representation.

Real-world scene analysis can benefit greatly not only from intensity information, bu- also from color, range, stereo, gradient, and texture information. In the past, scene analysis has been plagued by problems of noise, shadows, ambiguity, and variety. Many of these problems are much easier if information is available from several of these sources at the same time to produce redundant and complementary interpretations. This "multiple interpretation segmentation" is an important tool that has not really been exploited in any of the current scene-analysis programs.

C. REPRESENTATION

The internal representation of knowledge is a central issue in image understanding and problem solving. The overall success of a system heavily depends on the adequacy of the representation. There are two distinct forms of representation in scene analysis: geometric and symbolic. Geometric representations in scene analysis have been in terms of two-dimensional surfaces[56] and three-dimensional volumes.[45] The four symbolic representations that have been common are semantic nets[57] and procedural,[58] declarative,[59] and production systems.[60,61] The geometric representations model local order, derivable from primitive feature extraction, while the symbolic representations model global world order. A key issue in future real-world scene-analysis systems will be the

real-world scene-analysis systems will be the selection of adequate geometric and symbolic models and the intermediate conversion process between these two representations.

Although progress has been made in extending geometric two-dimensional surface representations to three dimensional volume representations,[48] as yet there is no general technique. Also, outdoor scenes pose their own problems in the geometric representation of natural shapes such as sky, clouds, and texture. For example, an adequate model for a complex natural object has never been constructed, even at the conceptual level, using any of these ideas. Unlike segmentation, real progress in this area is thus hampered by the lack of pure invention.

On the other hand, relatively general symbolic representations have been developed for world modeling. Currently, the declarative and procedural forms are receiving the most attention. The declarative form consists of a set of facts describing the knowledge and a collection of general rules (actually procedures) for manipulating facts. To solve a particular problem, a set of relevant facts (a knowledge domain) is manipulated until a success deduction is reached. In one declarative approach, called the state-space method,[62] the procedures are transformation rules and the deduction is a guided heuristic search that terminates when a goal is reached. In a second declarative approach, called theorem proving, facts are stated as axioms and the deduction is by formal proof procedures. The production system methodology developed in Section 2 uses a declarative form of knowledge representation. In this scheme, the facts or knowledge base is the information discovered during the feature-extraction process. The manipulation rules are a collection of condition/action productions that control the symbolic manipulation of this information. Such declarative representations are often inefficient for low-level vision unless some form of graph notation is used, but are quite natural for higher level vision.

The procedural form of representation is quite different. Procedural knowledge is stored (or embedded) within programs that either know or can compute the answer.[63] The motivation for procedural representations is that it is often valuable to associate control information about a fact with the fact itself.

The declarative form of knowledge representation has the advantage of easy modification by inserting or deleting axioms. Procedures, on the other hand, are modifiable only by the difficult process of debugging.[64] Declarative knowledge is also general purpose, while procedures tend to be special-purpose. Finally, the declarative form is more efficient and can easily integrate heuristic, semantic, and temporal knowledge.

Although procedural representations have been introduced in graphics,[65] they have only recently been utilized in scene analysis.[6] The extension of production systems with meta-rules for guidance is one way to give a declarative production system some of the advantages of a procedural representation.[32] Another possibility (described in Section 2) is the actual construction of procedures that make high-level use of visual information.

The third symbolic representation is the semantic net,[66] which consists of nodes corresponding to objects of surfaces and links corresponding to relations. The semantic net has found wide use in past scene-analysis work because there is a relatively natural correspondence to geometric models, and it allows simple deductions to be made trivially. Although nets (or graphs) can easily model spatial relations by themselves, they can neither easily represent temporal events nor specify how the resulting deductions are to be applied.[63]

Another important problem associated with representations is their interface to the rest of the system. In many areas of artificial intelligence there is an interface problem at the numeric/symbolic level. In scene analysis there is the additional barrier at the geometric (spatial)/symbolic (semantic) level. One interface mechanism for the geometric-symbolic level, is the use of graph rewriting rules operating in a production system framework (as described in Section 4). Such rules can specify the spatial relations of numeric operators, can be operated on themselves as symbolic entities, and can specify what numerical or symbolic form is to result. Such an extension was also recently made for semantic nets.[20]

D. CONTROL

The issue of control and system topology has received a great deal of attention in scene analysis.[67] Brief mention will be given here of the three principal structures: hierarchical, heterarchical, and production systems.

Early programs in scene analysis and artificial intelligence had a definite hierarchical control structure. Scenes were first processed with an edge detector, then a line finder, then a primitive matcher, and so on. The flow of control was from the bottom to the top.[56] Later, vision programs used model-directed or goal-guided search in which control also flowed from the top down.[4]

An alternative organization involves several subcomponents working on a problem simultaneously by passing information between them. This has been called a heterarchical organization. Heterarchy has been advocated as a cooperative method that could overcome some of the problems of linear organizations.[2] It allows components at all levels to exert goal-guided behavior without a vertical organization. Further, the control is distributed throughout all levels rather than only at the top level.

The most recent control scheme is called a production system.[61] In this form, knowledge is represented as an ordered set of rules (productions) consisting of a pattern and an action. If a pattern in the current data matches a production, then the action is executed, thus modifying the data.

A slightly modified production system has been developed for control of high- and low-level scene analysis. This work, described in Section 2, shows that production systems are interesting for real world scene analysis for several reasons. First, it is possible to embed a discrimination net in a production system.[68] Second, the productions themselves can behave as antecedent theorems as used in procedural forms.[63] Third, because the entire data set can be matched, a production can be triggered by global aspects, which is difficult in procedural representations.[63] And finally, production systems can provide the link between geometric and symbolic representations by incorporating graph

productions.[22] Generalized graph productions could act as geometric procedures to specify the spatial placement of primitive feature extraction operators.

E. DIRECTIONS FOR FUTURE RESEARCH

The principal limiting factor in outdoor scene analysis is the crude state of current representational capabilities. A good representation should be able to model natural shapes, texture, and three-dimensional information. If sufficiently rich representations for natural scenes were devised, then, rather than the present shotgun approach, there would be some direction to research in analysis techniques for the collection of the fundamental units.

Another important problem, at a more basic level, is the lack of a good implementation language for scene analysis. There is no existing language which has the following desirable features: efficient numeric computation, symbolic computation, clean syntax, basic scene analysis primitives, good debugging and editing facilities, reasonable portability, and good documentation. The development of such a language would dramatically affect the rate of progress, standardization of "working" modules, and exchange of capabilities between groups.

It is clear that parallelism will become an integral part of scene analysis systems, if only to achieve high throughput for the complex processes now evolving. Because of the crude state of current attempts, it is less clear whether or not parallelism will affect the methodology itself. The "parallel" algorithms that frequently find their way into publication show amazingly little creative effort to rise above the "micro" level of the problem. The few facilities that have even crude parallel processors (e.g., C.mmp) have, understandably and unfortunately, been bogged down in upgrading the state of the art in operating systems and control. Therefore, a serious attempt from within the vision community, with the right perspective, could have a dramatic impact on the whole issue of parallelism. The approach should be toward the entire system, at all levels of knowledge, rather than, as in the past, attempts at constructing only piecemeal algorithms. The emphasis should not be

on hardware, for the hardware problems can all be solved to some level of satisfaction. The real effort should be concerned with programming languages for efficiently and effectively specifying representations, processes, and control. There are certainly many seeds in current multi-process, AI-language, relaxation-process, production-system, and network research. But these are all only crude starts.

SECTION 5

PUBLICATION LIST

1. "The Performance of Edge Operators on Images with Texture," B.L. Bullock, Technical Report, October 1974 (to be published in C.G.I.P.).

2. "Real World Scene Analysis in Perspective," B.L. Bullock, presented at National Conf. of ACM, October 1975.

3. "Unstructured Control Concepts in Scene Analysis," B.L. Bullock, Hughes Research Laboratories Report 497, June 1976 (presented at Proc. 8th Annual Southeastern Symposium on System Theory, University of Tennessee, 1976).

4. "Finding Structures in Outdoor Scenes," B.L. Bullock, S.A. Dudani, J.P. Stafsudd, and C.S. Clark, Hughes Research Laboratories Report 498, July 1976 (presented at Joint Workshop on PR and AI, Hyannis, Mass., June 1976, and published in Pattern Recognition and Artificial Intelligence, Academic Press, 1976).

5. "Footprints: A Representation for Restricted Motion in Outdoor Scenes," B.L. Bullock and S.A. Dudani, Hughes Research Laboratories Research Report 504, September 1976, and also presented at 3rd International Joint Conference on Pattern Recognition, San Diego, November 1976.

6. "The Necessity for a Theory of Specialized Vision," B.L. Bullock, presented at ONR Workshop on Computer Vision, University of Mass., 1977 (to appear in Computer Vision, E. Riseman, ed., Academic Press, 1978).

7. AFOSR Interim ("Unstructured Control and Communication Process in Real World Scene Analysis"), October 1975.

8. AFOSR Interim ("Unstructured Control and Communication Process in Real World Scene Analysis"), October 1976.

REFERENCES

1. Proc. of the Workshop on Pattern-Directed Inference Systems, in SIGART Newsletter, No. 63, June 1977.

2. Y. Shirai, "A Heterarchical Program for Recognition of Polyhedra," MIT, AI LAB, Memo 263. Also in The Psychology of Computer Vision, P. Winston, ed. (McGraw-Hill, 1975).

3. B. Bullock, "Unstructured Control Concepts in Scene Analysis," HRL Research Report 497, June 1976. Also in Proc. 8th Annual Southeastern Symp. on System Theory, University of Tenn. (1976).

4. G. Falk, "Computer Interpretation of Imperfect Line Data of a 3-D Scene," Stanford University, AIM-132, 1970. Also in Artificial Intelligence, vol. 3 (1972).

5. G. Freuder, "Computer Systems for Visual Recognition Using Active Knowledge," MIT, AI LAB, TR-345, June 1976.

6. B. Kuipers, "A Frame for Frames: Representing Knowledge for Recognition," in Representation and Understanding, D. Bobrow, ed. (Academic Press, 1975).

7. D. Waltz, "Understanding Scenes with Shadows," in The Psychology of Computer Vision (McGraw-Hill, 1975).

8. M. Minsky, "A Framework for Representing Knowledge," in Winston (see Ref. 2).

9. A. Rosenfeld, R. Hummel, and S. Zucker, "Scene Labelling by Relaxation Operations," IEEE Trans. on SMC, SMC-16 (1976).

10. Y. Yakimovsky, "Scene Analysis Using a Semantic Base for Region Growing," Stanford University, A.I. LAB, AIM-209, June 1973.

11. M. Tenenbaum, "On Locating Objects by Their Distinguishing Features," Tech. Note 84, Stanford Research Institute, Sept. 1973.

12. B. Bullock, "Finding Structure in Outdoor Scenes," HRL Research Report 498, July 1976. Also in Pattern Recognition and Artificial Intelligence (Academic Press, 1976).

13. R. Ohlander, "Analysis of Natural Scenes," Ph.D. Thesis, Carnegie Mellon Univ., April 1975.

14. R. Bajcsy and L. Lieberman, "Computer Description of Real Outdoor Scenes," Proc. Second Joint Conf. on Pattern Recognition, Copenhagen, Aug. 1974.

15. B. Bullock, "Footprints: A Representation for Restricted Motion in Outdoor Scenes," HRL Research Report 504, Sept. 1976. Also in Proc. Third Int. Joint Conf. on Pattern Recognition, San Diego, November 1976.

16. R. Davis and J. King, "An Overview of Production Systems," Stanford University, A.I. LAB, Memo-271, October 1975.

17. T. Moran, "The Symbolic Imagery Hypothesis: A Production System Model," Ph.D. Thesis, Carnegie Mellon Univ., 1973.

18. E. Shortliff, Mycin: A Computer Based Medical Consultant (American-Elsevier, 1976).

19. R. Anderson, "The Use of Production Systems in RITA to Construct Personal Computer Agents," Proc. Workshop on Pattern Directed Inference, see Ref. 1.

20. R. Duda, "Semantic Network Representations in Rule-Based Inference Systems," in Ref. 1.

21. R. Kirsch, "Computer Determination of the Constituent-Structure of Biological Images," Computers and Biomedical Research, Vol. 4 (1971).

22. R. Anderson, "A New Approach to Programming Man-Machine Interfaces," Rand Report R-876-ARPA, March 1972.

23. L. Krakauer, "Computer Analysis of Visual Properties of Curved Objects," MIT, MAC TR-82, May 1971.

24. M. Baird and M. Kelly, "Recognizing Objects by Rules of Inference on Sequentially Thresholded Pictures," Computer Graphics and Image Processing, Vol. 3, March 1973.

25. R. Bajcsy and M. Tavakoli, "Image-Filtering — A Context-Dependent Process," IEEE Trans. on Circuits and Systems, Vol. CAS-22, No. 5, May 1975.

26. B. Buchanan, "Heuristic Dendral: A Program for Generating Explanatory Hypotheses in Organic Chemistry," in Machine Intelligence 4, B. Meltzer, ed. (American Elsevier, 1969).

27. R. Reddy, "The Hearsay Speech Understanding System," in Proc. Third IJCAI, August 1973.

28. S. Tanimoto, "An Iconic/Symbolic Data Structuring Scheme," in Artificial Intelligence and Pattern Recognition (Academic Press, 1977).

29. A. Kay, "Smalltalk Manual," Xerox Parc (1976).

30. D. Lenat, "Beings: Knowledge as Interacting Experts," Proc. 4th International Joint Conf. on Artificial Intelligence, Tbilisi, Georgian SSR, USSR (1975).

31. C. Hewitt and B. Smith, "Towards a Programming Apprentice," Proc. IEEE Trans. on Software Eng., SE-1, March 1975.

32. R. Davis, "The Application of Meta Level Knowledge in the Maintenance of Large Knowledge Bases," Stanford University, AIM-271, July 1976.

33. B. Bullock, "The Performance of Edge Operations on Images with Texture," HRL Tech. Report, Oct. 1974.

34. R. Duda and P. Hart, Pattern Classification and Scene Analysis (Wiley, 1973).

35. M. Hueckel, "A Local Visual Operator which Recognizes Edges and Lines," Journal of the ACM, Vol. 20, No. 4, October 1973.

36. R. Duda and P. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, No. 1, January 1972.

37. D. Ballard and J. Sklansky, "Tumor Detection in Radiographs," Computers and Biomedical Research, Vol. 6 (1973).

38. H. Moravec, Thesis Proposal, A.I. Lab., Stanford University, 1974.

39. S. Dudani and A. Luk, "Locating Straight-Line Edge Segments on Outdoor Scenes," in Proc. IEEE Conf. on Pattern Recognition and Image Processing," Troy, New York, June 1977.

40. A. Rosenfeld, Picture Processing by Computer (Academic Press, 1969).

41. U. Montanari, "On the Optimal Detection of Curves in Noisy Pictures," J. Assoc. for Computing Machinery, Vol. 14, 1971.

42. C. Brice and C. Fennema, "Scene Analysis Using Regions," Artificial Intelligence 1 (1970).

43. E. Carton, J. Weszka, J. Mohr, and A. Rosenfeld, "Some Basic Edge Detection Techniques," TR-277, University of Maryland, December 1973.

44. N. Gramenopoulos, "Terrain Type Recognition using ERTS-1 MSS Images," Symp. on Significant Results from the ERTS-1, NASA SP-327, Vol. 1, March 1973.

45. C. Agin, "Description and Representation of Curved Objects," Stanford A.I. Laboratory, AIM-174, October 1972.

46. R. Nevatia, "Motion Paralax Depth Ranging," Stanford A.I. Laboratory, memo in preparation.

47. R. Bajcsy, "Computer Identification of Textured Surfaces," Memo AIM-180, Stanford A.I. Laboratory, October 1972.

48. R. Nevatia and T. Binford, " Structured Descriptions of Complex Objects," Proc. 3rd Joint Conference on Artificial Intelligence, Stanford, August 1973.

49. R. Haralick and J. Bissell, "Texture-Tone Study with Application to Digitized Imagery," Tech. Rpt. 182-1, CRES, Lawrence, Kansas, December 1970.

50. D. Ausherman, S. Dwyer, and G. Lodwick, "Texture Discrimination within Digital Imagery," University of Missouri, IAL-TR 18-73, December 1972.

51. J. Gupta and P. Wintz, "A Boundary-Finding Algorithm and Its Applications," *IEEE Trans. on Circuits and Systems*, Vol. CAS-22, No. 14, April 1975.

52. E. Carton, J. Weszka, and A. Rosenfeld, "Some Basic Texture Analysis Techniques," TR-288, University of Maryland, January 1974.

53. F. Tomita, "Detection of Homogeneous Regions by Structural Analysis," Third Joint Conf. on Artificial Intelligence, August 1973.

54. E.S. Deustch, "Texture Descriptions Using Neighborhood Information," Computer Graphics and Image Processing, Vol. I, 145-168 (1972).

55. G. Lendaris and G. Stanley, "Diffraction Pattern Sampling for Automatic Pattern Recognition," Proc. IEEE 58, February 1970.

56. A. Guzman, "Computer Recognition of 3-D Objects in a Visual Scene," MIT, AD-692-200 (1968).

57. M.R. Quillian, "The Teachable Language Comprehender," Communications of the ACM, Vol. 12 (1969) pp. 459-476.

58. C. Hewitt, "Descriptions and Theoretical Analysis of Planner," MIT, Tech. Rpt. AI-258 (1972).

59. J. McCarthy and P. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," B. Meltzer, ed, Machine Intelligence 4, Edinburgh (1969).

60. A. Newell and H. Simon, Human Problem Solving (Prentice Hall, 1972).

61. A. Newell, "Production Systems: Models of Control Structures," Visual Information Proc.

62. N. Neilsson, Problem Solving Methods in Artificial Intelligence (McGraw Hill, 1971).

63. T. Winograd, "Five Lectures on Artificial Intelligence," Stanford Artificial Intelligence Lab., Memo AIM-246, September 1974.

64. Sussman, G., A Computer Model of Skill Acquisition, American Elsevier, 1975.

65. D. Grossman, "Procedural Representations of 3-D Objects," IBM Journal Research and Dev., Nov. 1976.

66. P. Winston, "Learning Structural Descriptions From Examples," MIT Project MAC, Tech. Rpt. TR-76, September 1970.

67. P. Winston, "Progress in Vision and Robotics," MIT Artificial Intelligence Lab., Tech. Rpt. TR-281, May 1973.

68. D.A. Waterman, "Adaptive Production Systems," Carnegie-Mellon University, Dept. of Psychology, Working Paper 285, December 1974.